SUMARI ANNEX

1. Codi del programa µC .......................................................... 2

2. Codi del programa Transferència .................................... 114

3. Plànols ................................................................................. 120
   3.1 Esquema de connexió .................................................. 120
   3.2 Disseny encapsulat ....................................................... 121

4. Taules Injecció ..................................................................... 122
   4.1 Valors Standards ......................................................... 122
   4.2 Valors Modificats ......................................................... 123

5. Fitxa dels components ....................................................... 124
   5.1 Transistor BD245C ....................................................... 124
   5.2 Transistor TIP142 ........................................................ 131
   5.3 Regulador LM7805 ....................................................... 136
   5.4 Regulador LM7809 ....................................................... 136
   5.5 Arduino Mega1280 ....................................................... 165
1. Codi del programa µC

// LLIBRERIES
#include <EEPROM.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(41, 39, 35, 33, 31);

// VARIABLES PROGRAMA
unsigned int buttonState0 = 0, lastbuttonState0 = 0, MODE = EEPROM.read(4000);
unsigned int buttonState1 = 0, lastButtonState1 = 0, SELECT = 0;  // Variables switch polsador SELECT

// VARIABLES CÀRREGA INJECCIÓ MODIFICADA DESDE LA EEPROM
unsigned int ii=0, hh=0, LOADED=0;

word value;
byte RHB,RLB;

// Variables Programa Funcionament CONTROL MOTOR
int rpm = 0;  // Valor RPM
int pulses = 0;  // nº polsos per llegir RPM
int hallsensor = 2;  // Sensor ciguenyal al PIN 2
float IgnitionTime;  // Variable temps retard xispa
float PreIgnitionTime;  // Valor calculat per trobar tim3
float IgnitionLoad;
int PulseCounter = 0;  // Contador cada 2 polsos per funcionament simultani
int AnalogPosition = 4;  // lectura sensor posició TPS
float TPS = 0;  // valor analògic de la papallona (Throttle position sensor)
int ValueAnalogPosition = 0;

unsigned int InjectionTime = 0;  // temps que carreguem al tcnt5(temps injecció)
//DECLARACIÓ ARRAYS(MATRÍUS)

int rpm9[41] = {
    58336, 56236, 55536, 54836, 54136, 53436, 52736, 52036, 51336, 50636, 49936, 49236, 48536, 47836, 47136, 46436, 45736, 45036, 44336, 42936, 41536, 40136, 38736, 37336, 35936, 34536, 33136, 31736, 30336, 28936, 27536, 26136, 24736, 23336, 21936, 20536, 19136, 17736, 16336, 14936, 13536};

int rpm10[41] = {
    60336, 58146, 56866, 55566, 55266, 54966, 53666, 53066, 52366, 51776, 50846, 50116, 49386, 48656, 47926, 47196, 46466, 45736, 44916, 44186, 41356, 39896, 38436, 36976, 36216, 34596, 33166, 31736, 31006, 29576, 28166, 26756, 25346, 23936, 22376, 20916, 19456, 17996, 16536, 15076, 13616};

int rpm11[41] = {
    60336, 58146, 56866, 55566, 55266, 54966, 53666, 53066, 52366, 51776, 50846, 50116, 49386, 48656, 47926, 47196, 46466, 45736, 44916, 44186, 41356, 39896, 38436, 36976, 36216, 34596, 33166, 31736, 31006, 29576, 28166, 26756, 25346, 23936, 22376, 20916, 19456, 17996, 16536, 15076, 13616};

int rpm12[41] = {
    62136, 59856, 59096, 58336, 57576, 56816, 56056, 55296, 54536, 53776, 53016, 52256, 51496, 50736, 49976, 49216, 48456, 47696, 46936, 46176, 45416, 44896, 44276, 43656, 43036, 42416, 41796, 40256, 39636, 39016, 38396, 37776, 37156, 36536, 35916, 35296, 34676, 34056, 33436, 32816, 32196, 31576, 30956, 30336, 29716, 29106, 28486, 27866, 27246, 26626, 26006, 25386, 24766, 24146, 23526, 22906, 22286, 21666, 21046, 20426, 19806, 19186, 18566, 17946, 17326, 16706, 16086, 15466, 14846, 14226, 13606, 12986, 12366, 11746, 11126, 10506, 9886, 9266, 8646, 8026, 7406, 6786, 6166, 5546, 4926, 4306, 3686, 3066, 2446, 1826, 1206, 586, 0;}

int rpm20[41] = {
    62736, 60426, 59656, 58886, 58116, 57346, 56576, 55806, 55036, 54266, 53496, 52726, 51956, 51186, 50416, 49646, 48876, 48106, 47336, 46566, 45796, 45026, 44256, 43486, 42716, 41946, 41176, 39406, 38636, 36966, 35296, 34526, 33756, 33086, 32416, 31746, 31076, 29406, 28736, 28066, 27396, 26726, 26056, 25386, 24716, 24046, 23376, 22706, 22036, 21366, 20696, 20026, 19356, 18686, 18016, 17346, 16676, 16006, 15336, 14666, 14006, 13336, 12666, 11996, 11326, 10656, 9986, 9316, 8646, 7976, 7306, 6636, 5966, 5296, 4626, 3956, 3286, 2616, 1946, 1276, 606, 0;}
int rpm27[41] = {
    65336, 62906, 62096, 61286, 60476, 59666, 58856, 58046, 57236, 56426, 55616,
    54806, 53996, 53186, 52376, 51566, 50756, 49946, 49136, 47516, 45896, 44276,
    42656, 41036, 39416, 37796, 36176, 34556, 32936, 31316, 29696, 28076, 26456,
    24836, 23216, 21596, 19976, 18356, 16736, 15116, 13496};

int rpm29[41] = {
    65536, 63676, 62856, 62036, 61216, 60396, 59576, 58756, 57936, 57116, 56296,
    55476, 54656, 53836, 53016, 52196, 51376, 50556, 49736, 48916, 48096, 46456, 44816,
    43176, 41536, 39896, 38256, 36616, 34976, 33336, 31696, 30056, 28416, 26776,
    25136, 23496, 21856, 20216, 18576, 16936, 15296, 13656};

int rpm32[41] = {
    65536, 64246, 63416, 62586, 61756, 60926, 60096, 59266, 58436, 57606, 56776,
    55946, 55116, 54286, 53456, 52626, 51796, 50966, 50136, 48476, 46816, 45156,
    43496, 41836, 40176, 38516, 36856, 35196, 33536, 31876, 30216, 28556, 26896,
    25236, 23576, 21916, 20256, 18596, 16936, 15296, 13656};

int rpm34[41] = {
    65536, 64816, 63976, 63136, 62296, 61456, 60616, 59776, 58936, 58096, 57256,
    56416, 55576, 54736, 53896, 53056, 52216, 51376, 50536, 48856, 47176,
    45496, 43816, 42136, 40456, 38776, 37096, 35416, 33736, 32056, 30376, 28696,
    27016, 25336, 23656, 21976, 20296, 18616, 16936, 15256, 13576};

int rpm36[41] = {
    65536, 65386, 64536, 63686, 62836, 61986, 61136, 60286, 59436, 58586, 57736,
    56886, 56036, 55186, 54336, 53486, 52636, 51786, 50936, 49096, 47236, 45386,
    44106, 42436, 40736, 39036, 37336, 35636, 33936, 32236, 30536, 28836, 27136,
    25436, 23736, 22036, 20336, 18636, 16936, 15236, 13536};

int rpm38[41] = {
    65536, 65536, 65536, 64986, 64116, 63246, 62376, 61506, 60636, 59766, 58896,
    58026, 57156, 56286, 55416, 54546, 53676, 52806, 51936, 50196, 48456,
    46716, 44976, 43236, 41496, 39756, 38016, 36276, 34536, 32796, 31056, 29316,
    27576, 25836, 24096, 22356, 20616, 18876, 17136, 15396, 13656};

int rpm40[41] = {
    65536, 65536, 65536, 65336, 64456, 63576, 62696, 61816, 60936, 60056,
    59176, 58296, 57416, 56536, 55656, 54776, 53896, 53016, 52136, 50236, 48366,
    46856, 45096, 43336, 41576, 39816, 38056, 36296, 34536, 32776, 31016, 29256,
    27496, 25736, 23976, 22216, 20456, 18696, 16936, 15176, 13416};
int rpm42[41] = {
  65536, 65536, 65536, 65536, 65196, 64306, 63416, 62526, 61636, 60746,
  59856, 58966, 58076, 57186, 56296, 55406, 54516, 53626, 52736, 50956, 49176,
  47396, 45616, 43836, 42056, 40276, 38496, 36716, 34936, 33156, 31376, 29596,
  27816, 26036, 24256, 22476, 20696, 18916, 17136, 15356, 13576};

int rpm43[41] = {
  65536, 65536, 65536, 65536, 65196, 64306, 63416, 62526, 61636, 60746,
  59856, 58966, 58076, 57186, 56296, 55406, 54516, 53626, 52736, 50956, 49176,
  47396, 45616, 43836, 42056, 40276, 38496, 36716, 34936, 33156, 31376, 29596,
  27816, 26036, 24256, 22476, 20696, 18916, 17136, 15356, 13576};

int rpm45[41] = {
  65536, 65536, 65536, 65536, 65196, 64306, 63416, 62526, 61636, 60746,
  59856, 58966, 58076, 57186, 56296, 55406, 54516, 53626, 52736, 50956, 49176,
  47396, 45616, 43836, 42056, 40276, 38496, 36716, 34936, 33156, 31376, 29596,
  27816, 26036, 24256, 22476, 20696, 18916, 17136, 15356, 13576};

int rpm48[41] = {
  65536, 65536, 65536, 65536, 65196, 64306, 63416, 62526, 61636, 60746,
  59856, 58966, 58076, 57186, 56296, 55406, 54516, 53626, 52736, 50956, 49176,
  47396, 45616, 43836, 42056, 40276, 38496, 36716, 34936, 33156, 31376, 29596,
  27816, 26036, 24256, 22476, 20696, 18916, 17136, 15356, 13576};

int rpm52[41] = {
  65536, 65536, 65536, 65536, 65196, 64306, 63416, 62526, 61636, 60746,
  59856, 58966, 58076, 57186, 56296, 55406, 54516, 53626, 52736, 50956, 49176,
  47396, 45616, 43836, 42056, 40276, 38496, 36716, 34936, 33156, 31376, 29596,
  27816, 26036, 24256, 22476, 20696, 18916, 17136, 15356, 13576};

// VARIABLES TRANSFERENCIA

unsigned int VAL, IND;   // VAL= Valor Llegit Port Sèrie. || IND= Valor índex trama
Port Sèrie

char buffer[5] = { 0,0,0,0,0};   // Punter VAL de 5 characters arribat per Port Sèrie.

char buff[3] = { 0,0,0};   // Punter IND de 3 characters arribat per Port Sèrie.

char ack=0x46;   // Byte de reconeixement trama correcte.
unsigned int gg=0, ff=0;      // Variables utilizades per increments interns de programa.
byte WHB,WLB;                  // WHB= Byte Hight Write. || WLB= Byte Low Write.

void setup() {
    pinMode(hallsensor, INPUT);      // Pin2 per Interrupció(0) Pujada
    pinMode(3, INPUT);               // Pin3 per Interrupcio(1) Pujada //Potser es pot borrar
    Serial.begin(19200);
    lcd.begin(16, 2);                // set up the LCD's number of columns and rows:
    attachInterrupt(0, RPM, RISING);  // Configuració interrupcio(0) RPM al PIN 2

    //Temporitzador 1 per Revolucions
    TIMSK1=0x01;                     // enabled global and timer overflow interrupt;
    TCCR1A = 0x00;
    TCNT1=0x796;                     // 16bit counter register
    TCCR1B = 0x04;                    // start timer/ set clock

    //Temporitzador 2 per generar PWM pel control del ralentí
    TCCR2B = 0x04;                   // definim freqüència

    //Temporitzador 3 per encesa
    TCCR3A = 0x00;
    TCNT3=0x0000;
    TCCR3B = 0x02;

    //Temporitzador 4 per contar Sms generació Espurna
    TCCR4A = 0x00;
    TCNT4=0xD8F0;
    TCCR4B = 0x02;

    //Temporitzador 5 per AMPLE POLS INJECCIÓ
    TIMSK5=0x01;
    TCCR5A = 0x00;
    TCNT5=0x0000
TCCR5B = 0x02
}
void loop () {
  lcd.setCursor(0, 0);    //Configuració LCD
  lcd.print(MODE);
  buttonState0 = digitalRead(3);
  if (buttonState0 != lastbuttonState0){
    if (buttonState0 == 1){
      MODE++;
      if(MODE>2)MODE=0;
    }
    lastbuttonState0 = buttonState0;
  }

  buttonState1 = digitalRead(4);
  if (buttonState1 != lastButtonState1){
    if (buttonState1 == 1){
      SELECT++;
      if(SELECT>1)SELECT=0;
    }
    lastButtonState1 = buttonState1;
  }
  //MODES DE FUNCIONAMENT
  switch (MODE) {
    //---------------------------------------------------------------------------------------
    //-------------------------------
    //---
    case 0:
lcd.setCursor(1, 0);
lcd.print(")TRANSFER MODE?");
lcd.setCursor(0, 1);
lcd.print("    ");
if (SELECT==1){
    lcd.setCursor(1, 0);
    lcd.print(" PLEASE CONNECT");
    lcd.setCursor(0, 1);
    lcd.print(" HARD & SOFT   ");
    LOADED=0;
    MODE=3;
}
break;

//Primer Cas

    case 1:
        lcd.setCursor(1, 0);
        lcd.print(")EXPERT MODE? ");
        lcd.setCursor(0, 1);
        lcd.print("    ");

        if (SELECT==1){
            lcd.setCursor(1, 0);
            lcd.print(")EXPERT LOADED");
            lcd.setCursor(0, 1);
            lcd.print("    ");
            MODE=4;
        }
        break;
//Segon cas

case 2:
    lcd.setCursor(1, 0);
    lcd.print(")STANDARD MODE?");
    lcd.setCursor(0, 1);
    lcd.print("                ");
    if (SELECT==1){
        lcd.setCursor(1, 0);
        lcd.print(")              ");
        lcd.setCursor(0, 1);
        lcd.print(" STANDARD LOADED");
        MODE=5;
    }
    break;

//Tercer Cas

case 3:
    SELECT=0;
    EEPROM.write(4000,3);
    if (Serial.available() > 0) {   // when characters arrive over the serial port...
        delay(4);       // wait a bit for the entire message to arrive
        if(Serial.read()==0x41) {
            gg=0;
            ff=0;
            for(unsigned int w=0;w<3;w++){
                buff[ff]=Serial.read();
                ff++;
            }
        }
    }
IND = IND + (buff[1] - 48) * 10;
IND = IND + (buff[0] - 48) * 100;

for(unsigned int v=0;v<5;v++){
buffer[gg] = Serial.read();
gg++;
}

VAL = VAL + (buffer[2] - 48) * 100;
VAL = VAL + (buffer[1] - 48) * 1000;
VAL = VAL + (buffer[0] - 48) * 10000;

WHB = highByte(VAL); // Partim la dada en part alta i baixa, guardem a l'EEPROM i l'ajuntem amb word més avall per llegir-la i comprovar que sigui correcte.
WLB = lowByte(VAL);

IND = IND * 2; // GUARDAR Taula Standard a la EEPROM entre 2000 i 3555
EEPROM.write(IND, WHB); //Guardem a la EEPROM al registre
IND++;
EEPROM.write(IND, WLB); //Guardem a la EEPROM
if (Serial.read() == ack){
Serial.write(0x47);
LOADED++;
}
else {

}
Serial.write(0x48);  
LOADED--;  
}  
}  
if (LOADED==779){  
lcd.setCursor(1, 0);  
lcd.print("--TRANSFER-- ");  
lcd.setCursor(0, 1);  
lcd.print(" COMPLETED!!! ");  
LOADED=0;  
}  
break;  

//Quart Cas

// CARREGUÉM MODE EXPERT VALORS DESDE EEPROM!!! (Registre: 0000 fins 1557)

SELECT=0;
EEPROM.write(4000,4);
ii=0;
hh=0;
for (unsigned int y=0; y<40; y++){  
  RHB=EEPROM.read(ii);  
  ii++;  
  RLB=EEPROM.read(ii);  
  ii++;  
  value=word(RHB,RLB);  
  rpm9[hh]=value;  
  hh++;  
}
hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm10[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm11[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
}
rpm12[hh]=value;
hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm20[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm22[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm24[hh]=value;
    hh++;
}
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm24[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm27[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm29[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
ii++;
RLB=EEPROM.read(ii);
ii++;
value=word(RHB,RLB);
rpm32[hh]=value;
hh++;}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
rpm34[hh]=value;
hh++;}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
rpm36[hh]=value;
hh++;}

hh=0;
for (unsigned int y=0; y<40; y++){  
    RHB=EEPROM.read(ii);  
    ii++;  
    RLB=EEPROM.read(ii);  
    ii++;  
    value=word(RHB,RLB);  
    rpm38[hh]=value;  
    hh++;  
}

hh=0;  
for (unsigned int y=0; y<40; y++){  
    RHB=EEPROM.read(ii);  
    ii++;  
    RLB=EEPROM.read(ii);  
    ii++;  
    value=word(RHB,RLB);  
    rpm40[hh]=value;  
    hh++;  
}

hh=0;  
for (unsigned int y=0; y<40; y++){  
    RHB=EEPROM.read(ii);  
    ii++;  
    RLB=EEPROM.read(ii);  
    ii++;  
    value=word(RHB,RLB);  
    rpm42[hh]=value;  
    hh++;  
}
} 

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm43[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm45[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
rpm48[hh]=value;
hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm52[hh]=value;
    hh++;}

hh=0;
MODE = 6;
break;

//Cinquè Cas

case 5:                  // CARREGUEM MODE STANDARD VALORS DESDE EEPROM!!!
(Registre: 2000 fins 3557)
    SELECT=0;
    EEPROM.write(4000,5);
    ii=2000;
    hh=0;
    for (unsigned int y=0; y<40; y++){
        RHB=EEPROM.read(ii);
        ii++;
        RLB=EEPROM.read(ii);
        ii++;
value=word(RHB,RLB);
rpm9[hh]=value;
hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){  
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm10[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){  
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm11[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){  
    RHB=EEPROM.read(ii);
    ii++;

RLB=EEPROM.read(ii);
ii++;
value=word(RHB,RLB);
rpm12[hh]=value;
hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm20[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm22[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
RHB=EEPROM.read(ii);
ii++;
RLB=EEPROM.read(ii);
ii++;
value=word(RHB,RLB);
rpm24[hh]=value;
hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm27[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm29[hh]=value;
    hh++;
}
hh=0;
for (unsigned int y=0; y<40; y++){
RHB=EEPROM.read(ii);
ii++;
RLB=EEPROM.read(ii);
ii++;
value=word(RHB,RLB);
rpm32[hh]=value;
hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
RHB=EEPROM.read(ii);
ii++;
RLB=EEPROM.read(ii);
ii++;
value=word(RHB,RLB);
rpm34[hh]=value;
hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){
RHB=EEPROM.read(ii);
ii++;
RLB=EEPROM.read(ii);
ii++;
value=word(RHB,RLB);
rpm36[hh]=value;
hh++; 
}
hh=0;
for (unsigned int y=0; y<40; y++){ 
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm38[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){ 
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm40[hh]=value;
    hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){ 
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm40[hh]=value;
    hh++;
}
value=word(RHB,RLB);
rpm42[hh]=value;

hh++;
}

hh=0;

for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm43[hh]=value;
    hh++;
}

hh=0;

for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm45[hh]=value;
    hh++;
}

hh=0;

for (unsigned int y=0; y<40; y++){
    RHB=EEPROM.read(ii);
    ii++;

RLB=EEPROM.read(ii);
ii++;
value=word(RHB,RLB);
rpm48[hh]=value;
hh++;
}

hh=0;
for (unsigned int y=0; y<40; y++){  
    RHB=EEPROM.read(ii);
    ii++;
    RLB=EEPROM.read(ii);
    ii++;
    value=word(RHB,RLB);
    rpm52[hh]=value;
    hh++;
}

hh=0;
MODE = 6;
break;

//Sisè Cas

case 6:
    SELECT = 0;
    analogWrite(10,82);     //Generació pulsos PWM per control del ralentí.

    if (rpm != 0 ) {
        TIMSK1=0x01;
        TIMSK3=0x01;         // enabled global and timer overflow interrupt;
        TIMSK4=0x01;         // enabled global and timer overflow interrupt;
    }
//TIMSK5=0x01;

//Càlcul del tEspurna

IgnitionTime = 1000*(0.058*exp(-0.001*rpm));
PreIgnitionTime = (IgnitionTime*65536)/32.768;
IgnitionLoad=65536-PreIgnitionTime;
ValueAnalogPosition = analogRead(AnalogPosition);
TPS = ((ValueAnalogPosition * 4.9)/1023);

//CONDICIONS PER DETERMINAR TEMPS D'INJECCIÓ QUE AGAFAREM DE LES ARRAYS

if ( rpm < 1000) {    //PER REVOLUCIONS INFERIOR A 1000
    if (TPS >=4){
        InjectionTime = rpm9[0];
    }
    else if(TPS >=3.85){
        InjectionTime = rpm9[1];
    }
    else if(TPS >=3.8){
        InjectionTime = rpm9[2];
    }
    else if(TPS >=3.75){
        InjectionTime = rpm9[3];
    }
    else if(TPS >=3.7){
        InjectionTime = rpm9[4];
    }
    else if(TPS >=3.65){
        InjectionTime = rpm9[5];
    }
}
else if(TPS >= 3.6)
    InjectionTime = rpm9[6];
}
else if(TPS >= 3.55)
    InjectionTime = rpm9[7];
}
else if(TPS >= 3.5)
    InjectionTime = rpm9[8];
}
else if(TPS >= 3.45)
    InjectionTime = rpm9[9];
}
else if(TPS >= 3.4)
    InjectionTime = rpm9[10];
}
else if(TPS >= 3.35)
    InjectionTime = rpm9[11];
}
else if(TPS >= 3.3)
    InjectionTime = rpm9[12];
}
else if(TPS >= 3.25)
    InjectionTime = rpm9[13];
}
else if(TPS >= 3.2)
    InjectionTime = rpm9[14];
}
else if(TPS >= 3.15){
InjectionTime = rpm9[15];
}
else if(TPS >= 3.1) {
    InjectionTime = rpm9[16];
}
else if(TPS >= 3.05) {
    InjectionTime = rpm9[17];
}
else if(TPS >= 3) {
    InjectionTime = rpm9[18];
}
else if(TPS >= 2.9) {
    InjectionTime = rpm9[19];
}
else if(TPS >= 2.8) {
    InjectionTime = rpm9[20];
}
else if(TPS >= 2.7) {
    InjectionTime = rpm9[21];
}
else if(TPS >= 2.6) {
    InjectionTime = rpm9[22];
}
else if(TPS >= 2.5) {
    InjectionTime = rpm9[23];
}
else if(TPS >= 2.4) {
    InjectionTime = rpm9[24];
} else if(TPS >= 2.3) {
    InjectionTime = rpm9[25];
}
else if(TPS >= 2.2) {
    InjectionTime = rpm9[26];
}
else if(TPS >= 2.1) {
    InjectionTime = rpm9[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm9[28];
}
else if(TPS >= 1.9) {
    InjectionTime = rpm9[29];
}
else if(TPS >= 1.8) {
    InjectionTime = rpm9[30];
}
else if(TPS >= 1.7) {
    InjectionTime = rpm9[31];
}
else if(TPS >= 1.6) {
    InjectionTime = rpm9[32];
}
else if(TPS >= 1.5) {
    InjectionTime = rpm9[33];
}
else if(TPS >= 1.4)
    InjectionTime = rpm9[34];
}
else if(TPS >= 1.3)
    InjectionTime = rpm9[35];
}
else if(TPS >= 1.2)
    InjectionTime = rpm9[36];
}
else if(TPS >= 1.1)
    InjectionTime = rpm9[37];
}
else if(TPS >= 1)
    InjectionTime = rpm9[38];
}
else if(TPS >= 0.9)
    InjectionTime = rpm9[39];
}
else if(TPS >= 0.8)
    InjectionTime = rpm9[40];
}
}
if ( 1000 < rpm && rpm < 1320) {
    if (TPS >= 4)
        InjectionTime = rpm10[0];
    }
else if(TPS >= 3.85)
    InjectionTime = rpm10[1];
    }
```c
} else if(TPS >= 3.8){
    InjectionTime = rpm10[2];
}
else if(TPS >= 3.75){
    InjectionTime = rpm10[3];
}
else if(TPS >= 3.7){
    InjectionTime = rpm10[4];
}
else if(TPS >= 3.65){
    InjectionTime = rpm10[5];
}
else if(TPS >= 3.6){
    InjectionTime = rpm10[6];
}
else if(TPS >= 3.55){
    InjectionTime = rpm10[7];
}
else if(TPS >= 3.5){
    InjectionTime = rpm10[8];
}
else if(TPS >= 3.45){
    InjectionTime = rpm10[9];
}
else if(TPS >= 3.4){
    InjectionTime = rpm10[10];
}
```

else if(TPS >= 3.35){
    InjectionTime = rpm10[11];
}
else if(TPS >= 3.3){
    InjectionTime = rpm10[12];
}
else if(TPS >= 3.25){
    InjectionTime = rpm10[13];
}
else if(TPS >= 3.2){
    InjectionTime = rpm10[14];
}
else if(TPS >= 3.15){
    InjectionTime = rpm10[15];
}
else if(TPS >= 3.1){
    InjectionTime = rpm10[16];
}
else if(TPS >= 3.05){
    InjectionTime = rpm10[17];
}
else if(TPS >= 3){
    InjectionTime = rpm10[18];
}
else if(TPS >= 2.9){
    InjectionTime = rpm10[19];
}
else if(TPS >= 2.8){
InjectionTime = rpm10[20];
}
else if(TPS >= 2.7){
    InjectionTime = rpm10[21];
}
else if(TPS >= 2.6){
    InjectionTime = rpm10[22];
}
else if(TPS >= 2.5){
    InjectionTime = rpm10[23];
}
else if(TPS >= 2.4){
    InjectionTime = rpm10[24];
}
else if(TPS >= 2.3){
    InjectionTime = rpm10[25];
}
else if(TPS >= 2.2){
    InjectionTime = rpm10[26];
}
else if(TPS >= 2.1){
    InjectionTime = rpm10[27];
}
else if(TPS >= 2){
    InjectionTime = rpm10[28];
}
else if(TPS >= 1.9){
    InjectionTime = rpm10[29];
else if(TPS >= 1.8)
    InjectionTime = rpm10[30];
}
else if(TPS >= 1.7)
    InjectionTime = rpm10[31];
}
else if(TPS >= 1.6)
    InjectionTime = rpm10[32];
}
else if(TPS >= 1.5)
    InjectionTime = rpm10[33];
}
else if(TPS >= 1.4)
    InjectionTime = rpm10[34];
}
else if(TPS >= 1.3)
    InjectionTime = rpm10[35];
}
else if(TPS >= 1.2)
    InjectionTime = rpm10[36];
}
else if(TPS >= 1.1)
    InjectionTime = rpm10[37];
}
else if(TPS >= 1)
    InjectionTime = rpm10[38];
}
else if(TPS >=0.9){
    InjectionTime = rpm10[39];
}
else if(TPS >=0.8){
    InjectionTime = rpm10[40];
}

if (1320 < rpm && rpm < 1690) {
    if (TPS >=4){
        InjectionTime = rpm11[0];
    }
    else if(TPS >=3.85){
        InjectionTime = rpm11[1];
    }
    else if(TPS >=3.8){
        InjectionTime = rpm11[2];
    }
    else if(TPS >=3.75){
        InjectionTime = rpm11[3];
    }
    else if(TPS >=3.7){
        InjectionTime = rpm11[4];
    }
    else if(TPS >=3.65){
        InjectionTime = rpm11[5];
    }
    else if(TPS >=3.6){
        InjectionTime = rpm11[6];
} else if(TPS >= 3.55){
    InjectionTime = rpm11[7];
}

} else if(TPS >= 3.5){
    InjectionTime = rpm11[8];
}

} else if(TPS >= 3.45){
    InjectionTime = rpm11[9];
}

} else if(TPS >= 3.4){
    InjectionTime = rpm11[10];
}

} else if(TPS >= 3.35){
    InjectionTime = rpm11[11];
}

} else if(TPS >= 3.3){
    InjectionTime = rpm11[12];
}

} else if(TPS >= 3.25){
    InjectionTime = rpm11[13];
}

} else if(TPS >= 3.2){
    InjectionTime = rpm11[14];
}

} else if(TPS >= 3.15){
    InjectionTime = rpm11[15];
}
else if(TPS >= 3.1){
    InjectionTime = rpm11[16];
}
else if(TPS >= 3.05){
    InjectionTime = rpm11[17];
}
else if(TPS >= 3){
    InjectionTime = rpm11[18];
}
else if(TPS >= 2.9){
    InjectionTime = rpm11[19];
}
else if(TPS >= 2.8){
    InjectionTime = rpm11[20];
}
else if(TPS >= 2.7){
    InjectionTime = rpm11[21];
}
else if(TPS >= 2.6){
    InjectionTime = rpm11[22];
}
else if(TPS >= 2.5){
    InjectionTime = rpm11[23];
}
else if(TPS >= 2.4){
    InjectionTime = rpm11[24];
}
else if(TPS >= 2.3){
InjectionTime = rpm11[25];
}
else if(TPS >=2.2){
    InjectionTime = rpm11[26];
}
else if(TPS >=2.1){
    InjectionTime = rpm11[27];
}
else if(TPS >=2){
    InjectionTime = rpm11[28];
}
else if(TPS >=1.9){
    InjectionTime = rpm11[29];
}
else if(TPS >=1.8){
    InjectionTime = rpm11[30];
}
else if(TPS >=1.7){
    InjectionTime = rpm11[31];
}
else if(TPS >=1.6){
    InjectionTime = rpm11[32];
}
else if(TPS >=1.5){
    InjectionTime = rpm11[33];
}
else if(TPS >=1.4){
    InjectionTime = rpm11[34];
else if(TPS >=1.3){
    InjectionTime = rpm11[35];
}
else if(TPS >=1.2){
    InjectionTime = rpm11[36];
}
else if(TPS >=1.1){
    InjectionTime = rpm11[37];
}
else if(TPS >=1){
    InjectionTime = rpm11[38];
}
else if(TPS >=0.9){
    InjectionTime = rpm11[39];
}
else if(TPS >=0.8){
    InjectionTime = rpm11[40];
}

if ( 1690 < rpm && rpm < 2040) { //CONDICIÓ REVOLUCIONS ENTRE 1690 I 2040
    if (TPS >=4){
        InjectionTime = rpm12[0];
    }
    else if(TPS >=3.85){
        InjectionTime = rpm12[1];
    }
    else if(TPS >=3.8){

InjectionTime = rpm12[2];
}
else if(TPS >=3.75){
    InjectionTime = rpm12[3];
}
else if(TPS >=3.7){
    InjectionTime = rpm12[4];
}
else if(TPS >=3.65){
    InjectionTime = rpm12[5];
}
else if(TPS >=3.6){
    InjectionTime = rpm12[6];
}
else if(TPS >=3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >=3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >=3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >=3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >=3.35){
    InjectionTime = rpm12[11];
else if(TPS >= 3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >= 3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >= 3.2){
    InjectionTime = rpm12[14];
}
else if(TPS >= 3.15){
    InjectionTime = rpm12[15];
}
else if(TPS >= 3.1){
    InjectionTime = rpm12[16];
}
else if(TPS >= 3.05){
    InjectionTime = rpm12[17];
}
else if(TPS >= 3){
    InjectionTime = rpm12[18];
}
else if(TPS >= 2.9){
    InjectionTime = rpm12[19];
}
else if(TPS >= 2.8){
    InjectionTime = rpm12[20];
}
else if(TPS >= 2.7) {
    InjectionTime = rpm12[21];
}
else if(TPS >= 2.6) {
    InjectionTime = rpm12[22];
}
else if(TPS >= 2.5) {
    InjectionTime = rpm12[23];
}
else if(TPS >= 2.4) {
    InjectionTime = rpm12[24];
}
else if(TPS >= 2.3) {
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2) {
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1) {
    InjectionTime = rpm12[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9) {
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8) {

InjectionTime = rpm12[30];
}
else if(TPS >= 1.7){
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6){
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5){
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4){
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3){
    InjectionTime = rpm12[35];
}
else if(TPS >= 1.2){
    InjectionTime = rpm12[36];
}
else if(TPS >= 1.1){
    InjectionTime = rpm12[37];
}
else if(TPS >= 1){
    InjectionTime = rpm12[38];
}
else if(TPS >= 0.9){
    InjectionTime = rpm12[39];
}
if (TPS >= 0.8) {
    InjectionTime = rpm12[40];
}

if (2040 < rpm && rpm < 2250) { //CONDICIÓ REVOLUCIONS ENTRE 2040 I 2250
    if (TPS >= 4) {
        InjectionTime = rpm12[0];
    }
    else if (TPS >= 3.85) {
        InjectionTime = rpm12[1];
    }
    else if (TPS >= 3.8) {
        InjectionTime = rpm12[2];
    }
    else if (TPS >= 3.75) {
        InjectionTime = rpm12[3];
    }
    else if (TPS >= 3.7) {
        InjectionTime = rpm12[4];
    }
    else if (TPS >= 3.65) {
        InjectionTime = rpm12[5];
    }
    else if (TPS >= 3.6) {
        InjectionTime = rpm12[6];
    }
    else if (TPS >= 3.55) {
InjectionTime = rpm12[7];
}
else if(TPS >=3.5){
        InjectionTime = rpm12[8];
}
else if(TPS >=3.45){
        InjectionTime = rpm12[9];
}
else if(TPS >=3.4){
        InjectionTime = rpm12[10];
}
else if(TPS >=3.35){
        InjectionTime = rpm12[11];
}
else if(TPS >=3.3){
        InjectionTime = rpm12[12];
}
else if(TPS >=3.25){
        InjectionTime = rpm12[13];
}
else if(TPS >=3.2){
        InjectionTime = rpm12[14];
}
else if(TPS >=3.15){
        InjectionTime = rpm12[15];
}
else if(TPS >=3.1){
        InjectionTime = rpm12[16];
if(TPS >= 3.05){
    InjectionTime = rpm12[17];
}

else if(TPS >= 3){
    InjectionTime = rpm12[18];
}

else if(TPS >= 2.9){
    InjectionTime = rpm12[19];
}

else if(TPS >= 2.8){
    InjectionTime = rpm12[20];
}

else if(TPS >= 2.7){
    InjectionTime = rpm12[21];
}

else if(TPS >= 2.6){
    InjectionTime = rpm12[22];
}

else if(TPS >= 2.5){
    InjectionTime = rpm12[23];
}

else if(TPS >= 2.4){
    InjectionTime = rpm12[24];
}

else if(TPS >= 2.3){
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2) {
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1) {
    InjectionTime = rpm12[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9) {
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8) {
    InjectionTime = rpm12[30];
}
else if(TPS >= 1.7) {
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6) {
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5) {
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4) {
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3) {
\begin{verbatim}
InjectionTime = rpm12[35];
}
else if(TPS >=1.2){
    InjectionTime = rpm12[36];
}
else if(TPS >=1.1){
    InjectionTime = rpm12[37];
}
else if(TPS >=1){
    InjectionTime = rpm12[38];
}
else if(TPS >=0.9){
    InjectionTime = rpm12[39];
}
else if(TPS >=0.8){
    InjectionTime = rpm12[40];
}
}
if ( 2250 < rpm && rpm < 2490) {
    if (TPS >=4){
        InjectionTime = rpm12[0];
    }
    else if(TPS >=3.85){
        InjectionTime = rpm12[1];
    }
    else if(TPS >=3.8){
        InjectionTime = rpm12[2];
    }
}
\end{verbatim}
else if(TPS >= 3.75){
    InjectionTime = rpm12[3];
}
else if(TPS >= 3.7){
    InjectionTime = rpm12[4];
}
else if(TPS >= 3.65){
    InjectionTime = rpm12[5];
}
else if(TPS >= 3.6){
    InjectionTime = rpm12[6];
}
else if(TPS >= 3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >= 3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >= 3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >= 3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >= 3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >= 3.3){
```c
InjectionTime = rpm12[12];
}
else if(TPS >= 3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >= 3.2){
    InjectionTime = rpm12[14];
}
else if(TPS >= 3.15){
    InjectionTime = rpm12[15];
}
else if(TPS >= 3.1){
    InjectionTime = rpm12[16];
}
else if(TPS >= 3.05){
    InjectionTime = rpm12[17];
}
else if(TPS >= 3){
    InjectionTime = rpm12[18];
}
else if(TPS >= 2.9){
    InjectionTime = rpm12[19];
}
else if(TPS >= 2.8){
    InjectionTime = rpm12[20];
}
else if(TPS >= 2.7){
    InjectionTime = rpm12[21];
```
else if(TPS >= 2.6) {
    InjectionTime = rpm12[22];
}
else if(TPS >= 2.5) {
    InjectionTime = rpm12[23];
}
else if(TPS >= 2.4) {
    InjectionTime = rpm12[24];
}
else if(TPS >= 2.3) {
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2) {
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1) {
    InjectionTime = rpm12[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9) {
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8) {
    InjectionTime = rpm12[30];
}
else if(TPS >= 1.7) {
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6) {
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5) {
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4) {
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3) {
    InjectionTime = rpm12[35];
}
else if(TPS >= 1.2) {
    InjectionTime = rpm12[36];
}
else if(TPS >= 1.1) {
    InjectionTime = rpm12[37];
}
else if(TPS >= 1) {
    InjectionTime = rpm12[38];
}
else if(TPS >= 0.9) {
    InjectionTime = rpm12[39];
}
else if(TPS >= 0.8) {
InjectionTime = rpm12[40];
}
}
if (2490 < rpm && rpm < 2760) {//CONDICIÓ REVOLUCIONS ENTRE 2490 I 2760
    if (TPS >= 4){
        InjectionTime = rpm12[0];
    }
    else if(TPS >= 3.85){
        InjectionTime = rpm12[1];
    }
    else if(TPS >= 3.8){
        InjectionTime = rpm12[2];
    }
    else if(TPS >= 3.75){
        InjectionTime = rpm12[3];
    }
    else if(TPS >= 3.7){
        InjectionTime = rpm12[4];
    }
    else if(TPS >= 3.65){
        InjectionTime = rpm12[5];
    }
    else if(TPS >= 3.6){
        InjectionTime = rpm12[6];
    }
    else if(TPS >= 3.55){
        InjectionTime = rpm12[7];
    }
}
else if(TPS >= 3.5) {
    InjectionTime = rpm12[8];
}
else if(TPS >= 3.45) {
    InjectionTime = rpm12[9];
}
else if(TPS >= 3.4) {
    InjectionTime = rpm12[10];
}
else if(TPS >= 3.35) {
    InjectionTime = rpm12[11];
}
else if(TPS >= 3.3) {
    InjectionTime = rpm12[12];
}
else if(TPS >= 3.25) {
    InjectionTime = rpm12[13];
}
else if(TPS >= 3.2) {
    InjectionTime = rpm12[14];
}
else if(TPS >= 3.15) {
    InjectionTime = rpm12[15];
}
else if(TPS >= 3.1) {
    InjectionTime = rpm12[16];
}
else if(TPS >= 3.05) {
InjectionTime = rpm12[17];
}
else if(TPS >=3){
    InjectionTime = rpm12[18];
}
else if(TPS >=2.9){
    InjectionTime = rpm12[19];
}
else if(TPS >=2.8){
    InjectionTime = rpm12[20];
}
else if(TPS >=2.7){
    InjectionTime = rpm12[21];
}
else if(TPS >=2.6){
    InjectionTime = rpm12[22];
}
else if(TPS >=2.5){
    InjectionTime = rpm12[23];
}
else if(TPS >=2.4){
    InjectionTime = rpm12[24];
}
else if(TPS >=2.3){
    InjectionTime = rpm12[25];
}
else if(TPS >=2.2){
    InjectionTime = rpm12[26];
else if(TPS >= 2.1) {
    InjectionTime = rpm12[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9) {
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8) {
    InjectionTime = rpm12[30];
}
else if(TPS >= 1.7) {
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6) {
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5) {
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4) {
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3) {
    InjectionTime = rpm12[35];
}
else if(TPS >= 1.2) {
    InjectionTime = rpm12[36];
}
else if(TPS >= 1.1) {
    InjectionTime = rpm12[37];
}
else if(TPS >= 1) {
    InjectionTime = rpm12[38];
}
else if(TPS >= 0.9) {
    InjectionTime = rpm12[39];
}
else if(TPS >= 0.8) {
    InjectionTime = rpm12[40];
}

if (2760 < rpm && rpm < 2940) {
    if (TPS >= 4) {
        InjectionTime = rpm12[0];
    }
    else if(TPS >= 3.85) {
        InjectionTime = rpm12[1];
    }
    else if(TPS >= 3.8) {
        InjectionTime = rpm12[2];
    }
    else if(TPS >= 3.75) {
        InjectionTime = rpm12[3];
    }
}  
else if(TPS >= 3.7){  
    InjectionTime = rpm12[4];  
}  
else if(TPS >= 3.65){  
    InjectionTime = rpm12[5];  
}  
else if(TPS >= 3.6){  
    InjectionTime = rpm12[6];  
}  
else if(TPS >= 3.55){  
    InjectionTime = rpm12[7];  
}  
else if(TPS >= 3.5){  
    InjectionTime = rpm12[8];  
}  
else if(TPS >= 3.45){  
    InjectionTime = rpm12[9];  
}  
else if(TPS >= 3.4){  
    InjectionTime = rpm12[10];  
}  
else if(TPS >= 3.35){  
    InjectionTime = rpm12[11];  
}  
else if(TPS >= 3.3){  
    InjectionTime = rpm12[12];  
}  
}
else if(TPS >= 3.25) {
    InjectionTime = rpm12[13];
}
else if(TPS >= 3.2) {
    InjectionTime = rpm12[14];
}
else if(TPS >= 3.15) {
    InjectionTime = rpm12[15];
}
else if(TPS >= 3.1) {
    InjectionTime = rpm12[16];
}
else if(TPS >= 3.05) {
    InjectionTime = rpm12[17];
}
else if(TPS >= 3) {
    InjectionTime = rpm12[18];
}
else if(TPS >= 2.9) {
    InjectionTime = rpm12[19];
}
else if(TPS >= 2.8) {
    InjectionTime = rpm12[20];
}
else if(TPS >= 2.7) {
    InjectionTime = rpm12[21];
}
else if(TPS >= 2.6) {


InjectionTime = rpm12[22];
}
else if(TPS >=2.5){
    InjectionTime = rpm12[23];
}
else if(TPS >=2.4){
    InjectionTime = rpm12[24];
}
else if(TPS >=2.3){
    InjectionTime = rpm12[25];
}
else if(TPS >=2.2){
    InjectionTime = rpm12[26];
}
else if(TPS >=2.1){
    InjectionTime = rpm12[27];
}
else if(TPS >=2.0){
    InjectionTime = rpm12[28];
}
else if(TPS >=1.9){
    InjectionTime = rpm12[29];
}
else if(TPS >=1.8){
    InjectionTime = rpm12[30];
}
else if(TPS >=1.7){
    InjectionTime = rpm12[31];
} else if(TPS >= 1.6){
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5){
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4){
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3){
    InjectionTime = rpm12[35];
}
else if(TPS >= 1.2){
    InjectionTime = rpm12[36];
}
else if(TPS >= 1.1){
    InjectionTime = rpm12[37];
}
else if(TPS >= 1.0){
    InjectionTime = rpm12[38];
}
else if(TPS >= 0.9){
    InjectionTime = rpm12[39];
}
else if(TPS >= 0.8){
    InjectionTime = rpm12[40];
}
if (2940 < rpm && rpm < 3240) {  //PER REVOLUCIONS ENTRE 2940 3240
    if (TPS >= 4){
        InjectionTime = rpm12[0];
    }
    else if(TPS >= 3.85){
        InjectionTime = rpm12[1];
    }
    else if(TPS >= 3.8){
        InjectionTime = rpm12[2];
    }
    else if(TPS >= 3.75){
        InjectionTime = rpm12[3];
    }
    else if(TPS >= 3.7){
        InjectionTime = rpm12[4];
    }
    else if(TPS >= 3.65){
        InjectionTime = rpm12[5];
    }
    else if(TPS >= 3.6){
        InjectionTime = rpm12[6];
    }
    else if(TPS >= 3.55){
        InjectionTime = rpm12[7];
    }
    else if(TPS >= 3.5){
        InjectionTime = rpm12[8];
    }
else if(TPS >=3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >=3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >=3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >=3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >=3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >=3.2){
    InjectionTime = rpm12[14];
}
else if(TPS >=3.15){
    InjectionTime = rpm12[15];
}
else if(TPS >=3.1){
    InjectionTime = rpm12[16];
}
else if(TPS >=3.05){
    InjectionTime = rpm12[17];
}
else if(TPS >=3){
    InjectionTime = rpm12[18];
}
else if(TPS >=2.9){
    InjectionTime = rpm12[19];
}
else if(TPS >=2.8){
    InjectionTime = rpm12[20];
}
else if(TPS >=2.7){
    InjectionTime = rpm12[21];
}
else if(TPS >=2.6){
    InjectionTime = rpm12[22];
}
else if(TPS >=2.5){
    InjectionTime = rpm12[23];
}
else if(TPS >=2.4){
    InjectionTime = rpm12[24];
}
else if(TPS >=2.3){
    InjectionTime = rpm12[25];
}
else if(TPS >=2.2){
    InjectionTime = rpm12[26];
}
else if(TPS >=2.1){
InjectionTime = rpm12[27];
}
else if(TPS >=2){
    InjectionTime = rpm12[28];
}
else if(TPS >=1.9){
    InjectionTime = rpm12[29];
}
else if(TPS >=1.8){
    InjectionTime = rpm12[30];
}
else if(TPS >=1.7){
    InjectionTime = rpm12[31];
}
else if(TPS >=1.6){
    InjectionTime = rpm12[32];
}
else if(TPS >=1.5){
    InjectionTime = rpm12[33];
}
else if(TPS >=1.4){
    InjectionTime = rpm12[34];
}
else if(TPS >=1.3){
    InjectionTime = rpm12[35];
}
else if(TPS >=1.2){
    InjectionTime = rpm12[36];
else if(TPS >=1.1)
    InjectionTime = rpm12[37];
}
else if(TPS >=1)
    InjectionTime = rpm12[38];
}
else if(TPS >=0.9)
    InjectionTime = rpm12[39];
}
else if(TPS >=0.8)
    InjectionTime = rpm12[40];
}

if ( 3240 < rpm && rpm < 3450) {
  if (TPS >=4)
    InjectionTime = rpm12[0];
  }
else if(TPS >=3.85)
    InjectionTime = rpm12[1];
  }
else if(TPS >=3.8)
    InjectionTime = rpm12[2];
  }
else if(TPS >=3.75)
    InjectionTime = rpm12[3];
  }
else if(TPS >=3.7)
InjectionTime = rpm12[4];
}
else if(TPS >=3.65){
    InjectionTime = rpm12[5];
}
else if(TPS >=3.6){
    InjectionTime = rpm12[6];
}
else if(TPS >=3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >=3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >=3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >=3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >=3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >=3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >=3.25){
    InjectionTime = rpm12[13];
else if(TPS >= 3.2) {
    InjectionTime = rpm12[14];
}
else if(TPS >= 3.15) {
    InjectionTime = rpm12[15];
}
else if(TPS >= 3.1) {
    InjectionTime = rpm12[16];
}
else if(TPS >= 3.05) {
    InjectionTime = rpm12[17];
}
else if(TPS >= 3) {
    InjectionTime = rpm12[18];
}
else if(TPS >= 2.9) {
    InjectionTime = rpm12[19];
}
else if(TPS >= 2.8) {
    InjectionTime = rpm12[20];
}
else if(TPS >= 2.7) {
    InjectionTime = rpm12[21];
}
else if(TPS >= 2.6) {
    InjectionTime = rpm12[22];
}
else if(TPS >= 2.5) {
    InjectionTime = rpm12[23];
}
else if(TPS >= 2.4) {
    InjectionTime = rpm12[24];
}
else if(TPS >= 2.3) {
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2) {
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1) {
    InjectionTime = rpm12[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9) {
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8) {
    InjectionTime = rpm12[30];
}
else if(TPS >= 1.7) {
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6) {

InjectionTime = rpm12[32];
}
else if(TPS >=1.5){
    InjectionTime = rpm12[33];
}
else if(TPS >=1.4){
    InjectionTime = rpm12[34];
}
else if(TPS >=1.3){
    InjectionTime = rpm12[35];
}
else if(TPS >=1.2){
    InjectionTime = rpm12[36];
}
else if(TPS >=1.1){
    InjectionTime = rpm12[37];
}
else if(TPS >=1){
    InjectionTime = rpm12[38];
}
else if(TPS >=0.9){
    InjectionTime = rpm12[39];
}
else if(TPS >=0.8){
    InjectionTime = rpm12[40];
}
}

if( 3450 < rpm && rpm < 3600) {
if (TPS >=4){
    InjectionTime = rpm12[0];
}
else if(TPS >=3.85){
    InjectionTime = rpm12[1];
}
else if(TPS >=3.8){
    InjectionTime = rpm12[2];
}
else if(TPS >=3.75){
    InjectionTime = rpm12[3];
}
else if(TPS >=3.7){
    InjectionTime = rpm12[4];
}
else if(TPS >=3.65){
    InjectionTime = rpm12[5];
}
else if(TPS >=3.6){
    InjectionTime = rpm12[6];
}
else if(TPS >=3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >=3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >=3.45){
InjectionTime = rpm12[9];
}
else if(TPS >=3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >=3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >=3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >=3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >=3.2){
    InjectionTime = rpm12[14];
}
else if(TPS >=3.15){
    InjectionTime = rpm12[15];
}
else if(TPS >=3.1){
    InjectionTime = rpm12[16];
}
else if(TPS >=3.05){
    InjectionTime = rpm12[17];
}
else if(TPS >=3){
    InjectionTime = rpm12[18];
else if(TPS >= 2.9){
    InjectionTime = rpm12[19];
}
else if(TPS >= 2.8){
    InjectionTime = rpm12[20];
}
else if(TPS >= 2.7){
    InjectionTime = rpm12[21];
}
else if(TPS >= 2.6){
    InjectionTime = rpm12[22];
}
else if(TPS >= 2.5){
    InjectionTime = rpm12[23];
}
else if(TPS >= 2.4){
    InjectionTime = rpm12[24];
}
else if(TPS >= 2.3){
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2){
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1){
    InjectionTime = rpm12[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9) {
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8) {
    InjectionTime = rpm12[30];
}
else if(TPS >= 1.7) {
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6) {
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5) {
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4) {
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3) {
    InjectionTime = rpm12[35];
}
else if(TPS >= 1.2) {
    InjectionTime = rpm12[36];
}
else if(TPS >= 1.1) {

InjectionTime = rpm12[37];

} else if(TPS >=1){
    InjectionTime = rpm12[38];
}

else if(TPS >=0.9){
    InjectionTime = rpm12[39];
}

else if(TPS >=0.8){
    InjectionTime = rpm12[40];
}

}

if ( 3600 < rpm && rpm < 3800) { //PER REVOLUCIONS ENTRE 3600 I 3800
    if (TPS >=4){
        InjectionTime = rpm12[0];
    }
    else if(TPS >=3.85){
        InjectionTime = rpm12[1];
    }
    else if(TPS >=3.8){
        InjectionTime = rpm12[2];
    }
    else if(TPS >=3.75){
        InjectionTime = rpm12[3];
    }
    else if(TPS >=3.7){
        InjectionTime = rpm12[4];
    }
else if(TPS >= 3.65){
    InjectionTime = rpm12[5];
}
else if(TPS >= 3.6){
    InjectionTime = rpm12[6];
}
else if(TPS >= 3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >= 3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >= 3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >= 3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >= 3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >= 3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >= 3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >= 3.2){
    InjectionTime = rpm12[14];
}
InjectionTime = rpm12[14];

} else if(TPS >= 3.15) {
    InjectionTime = rpm12[15];
}

else if(TPS >= 3.1) {
    InjectionTime = rpm12[16];
}

else if(TPS >= 3.05) {
    InjectionTime = rpm12[17];
}

else if(TPS >= 3) {
    InjectionTime = rpm12[18];
}

else if(TPS >= 2.9) {
    InjectionTime = rpm12[19];
}

else if(TPS >= 2.8) {
    InjectionTime = rpm12[20];
}

else if(TPS >= 2.7) {
    InjectionTime = rpm12[21];
}

else if(TPS >= 2.6) {
    InjectionTime = rpm12[22];
}

else if(TPS >= 2.5) {
    InjectionTime = rpm12[23];
else if(TPS >= 2.4) {
    InjectionTime = rpm12[24];
}
else if(TPS >= 2.3) {
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2) {
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1) {
    InjectionTime = rpm12[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9) {
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8) {
    InjectionTime = rpm12[30];
}
else if(TPS >= 1.7) {
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6) {
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5) {
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4) {
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3) {
    InjectionTime = rpm12[35];
}
else if(TPS >= 1.2) {
    InjectionTime = rpm12[36];
}
else if(TPS >= 1.1) {
    InjectionTime = rpm12[37];
}
else if(TPS >= 1) {
    InjectionTime = rpm12[38];
}
else if(TPS >= 0.9) {
    InjectionTime = rpm12[39];
}
else if(TPS >= 0.8) {
    InjectionTime = rpm12[40];
}
}
if (3800 < rpm && rpm < 4080) {
    if (TPS >= 4) {
        InjectionTime = rpm12[0];
    }
if(TPS >= 3.85){
    InjectionTime = rpm12[1];
}
else if(TPS >= 3.8){
    InjectionTime = rpm12[2];
}
else if(TPS >= 3.75){
    InjectionTime = rpm12[3];
}
else if(TPS >= 3.7){
    InjectionTime = rpm12[4];
}
else if(TPS >= 3.65){
    InjectionTime = rpm12[5];
}
else if(TPS >= 3.6){
    InjectionTime = rpm12[6];
}
else if(TPS >= 3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >= 3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >= 3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >= 3.4) {
    InjectionTime = rpm12[10];
}
else if(TPS >= 3.35) {
    InjectionTime = rpm12[11];
}
else if(TPS >= 3.3) {
    InjectionTime = rpm12[12];
}
else if(TPS >= 3.25) {
    InjectionTime = rpm12[13];
}
else if(TPS >= 3.2) {
    InjectionTime = rpm12[14];
}
else if(TPS >= 3.15) {
    InjectionTime = rpm12[15];
}
else if(TPS >= 3.1) {
    InjectionTime = rpm12[16];
}
else if(TPS >= 3.05) {
    InjectionTime = rpm12[17];
}
else if(TPS >= 3) {
    InjectionTime = rpm12[18];
}
else if(TPS >= 2.9) {

InjectionTime = rpm12[19];
}
else if(TPS >= 2.8){
    InjectionTime = rpm12[20];
}
else if(TPS >= 2.7){
    InjectionTime = rpm12[21];
}
else if(TPS >= 2.6){
    InjectionTime = rpm12[22];
}
else if(TPS >= 2.5){
    InjectionTime = rpm12[23];
}
else if(TPS >= 2.4){
    InjectionTime = rpm12[24];
}
else if(TPS >= 2.3){
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2){
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1){
    InjectionTime = rpm12[27];
}
else if(TPS >= 2){
    InjectionTime = rpm12[28];
}
} 
else if(TPS >= 1.9){
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8){
    InjectionTime = rpm12[30];
}
else if(TPS >= 1.7){
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6){
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5){
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4){
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3){
    InjectionTime = rpm12[35];
}
else if(TPS >= 1.2){
    InjectionTime = rpm12[36];
}
else if(TPS >= 1.1){
    InjectionTime = rpm12[37];
}
else if(TPS >=1) {
    InjectionTime = rpm12[38];
}
else if(TPS >=0.9) {
    InjectionTime = rpm12[39];
}
else if(TPS >=0.8) {
    InjectionTime = rpm12[40];
}

if (4080 < rpm && rpm < 4200) {
    if (TPS >=4) {
        InjectionTime = rpm12[0];
    }
    else if(TPS >=3.85) {
        InjectionTime = rpm12[1];
    }
    else if(TPS >=3.8) {
        InjectionTime = rpm12[2];
    }
    else if(TPS >=3.75) {
        InjectionTime = rpm12[3];
    }
    else if(TPS >=3.7) {
        InjectionTime = rpm12[4];
    }
    else if(TPS >=3.65) {
        InjectionTime = rpm12[5];
    }
else if(TPS >=3.6){
    InjectionTime = rpm12[6];
}
else if(TPS >=3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >=3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >=3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >=3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >=3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >=3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >=3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >=3.2){
    InjectionTime = rpm12[14];
}
else if(TPS >=3.15){
    InjectionTime = rpm12[15];
}
else if(TPS >=3.1){
    InjectionTime = rpm12[16];
}
else if(TPS >=3.05){
    InjectionTime = rpm12[17];
}
else if(TPS >=3){
    InjectionTime = rpm12[18];
}
else if(TPS >=2.9){
    InjectionTime = rpm12[19];
}
else if(TPS >=2.8){
    InjectionTime = rpm12[20];
}
else if(TPS >=2.7){
    InjectionTime = rpm12[21];
}
else if(TPS >=2.6){
    InjectionTime = rpm12[22];
}
else if(TPS >=2.5){
    InjectionTime = rpm12[23];
}
else if(TPS >=2.4){

\text{InjectionTime} = \text{rpm12}[24];

}  
else if(\text{TPS} \geq 2.3)\
   \text{InjectionTime} = \text{rpm12}[25];

}  
else if(\text{TPS} \geq 2.2)\
   \text{InjectionTime} = \text{rpm12}[26];

}  
else if(\text{TPS} \geq 2.1)\
   \text{InjectionTime} = \text{rpm12}[27];

}  
else if(\text{TPS} \geq 2)\
   \text{InjectionTime} = \text{rpm12}[28];

}  
else if(\text{TPS} \geq 1.9)\
   \text{InjectionTime} = \text{rpm12}[29];

}  
else if(\text{TPS} \geq 1.8)\
   \text{InjectionTime} = \text{rpm12}[30];

}  
else if(\text{TPS} \geq 1.7)\
   \text{InjectionTime} = \text{rpm12}[31];

}  
else if(\text{TPS} \geq 1.6)\
   \text{InjectionTime} = \text{rpm12}[32];

}  
else if(\text{TPS} \geq 1.5)\
   \text{InjectionTime} = \text{rpm12}[33];
else if(TPS >=1.4){
    InjectionTime = rpm12[34];
}
else if(TPS >=1.3){
    InjectionTime = rpm12[35];
}
else if(TPS >=1.2){
    InjectionTime = rpm12[36];
}
else if(TPS >=1.1){
    InjectionTime = rpm12[37];
}
else if(TPS >=1){
    InjectionTime = rpm12[38];
}
else if(TPS >=0.9){
    InjectionTime = rpm12[39];
}
else if(TPS >=0.8){
    InjectionTime = rpm12[40];
}

if ( 4200 < rpm && rpm < 4320) {
    if (TPS >=4){
        InjectionTime = rpm12[0];
    }
    else if(TPS >=3.85){

InjectionTime = rpm12[1];
}
else if(TPS >= 3.8){
    InjectionTime = rpm12[2];
}
else if(TPS >= 3.75){
    InjectionTime = rpm12[3];
}
else if(TPS >= 3.7){
    InjectionTime = rpm12[4];
}
else if(TPS >= 3.65){
    InjectionTime = rpm12[5];
}
else if(TPS >= 3.6){
    InjectionTime = rpm12[6];
}
else if(TPS >= 3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >= 3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >= 3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >= 3.4){
    InjectionTime = rpm12[10];
else if(TPS >=3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >=3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >=3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >=3.2){
    InjectionTime = rpm12[14];
}
else if(TPS >=3.15){
    InjectionTime = rpm12[15];
}
else if(TPS >=3.1){
    InjectionTime = rpm12[16];
}
else if(TPS >=3.05){
    InjectionTime = rpm12[17];
}
else if(TPS >=3){
    InjectionTime = rpm12[18];
}
else if(TPS >=2.9){
    InjectionTime = rpm12[19];
}
else if(TPS >= 2.8) {
    InjectionTime = rpm12[20];
}
else if(TPS >= 2.7) {
    InjectionTime = rpm12[21];
}
else if(TPS >= 2.6) {
    InjectionTime = rpm12[22];
}
else if(TPS >= 2.5) {
    InjectionTime = rpm12[23];
}
else if(TPS >= 2.4) {
    InjectionTime = rpm12[24];
}
else if(TPS >= 2.3) {
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2) {
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1) {
    InjectionTime = rpm12[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9) {
InjectionTime = rpm12[29];
}
else if(TPS >= 1.8) {
    InjectionTime = rpm12[30];
}
else if(TPS >= 1.7) {
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6) {
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5) {
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4) {
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3) {
    InjectionTime = rpm12[35];
}
else if(TPS >= 1.2) {
    InjectionTime = rpm12[36];
}
else if(TPS >= 1.1) {
    InjectionTime = rpm12[37];
}
else if(TPS >= 1) {
    InjectionTime = rpm12[38];
else if(TPS >=0.9){
    InjectionTime = rpm12[39];
}

else if(TPS >=0.8){
    InjectionTime = rpm12[40];
}

if ( 4320 < rpm && rpm < 4500) {  //PER REVOLUCIONS ENTRE 4320 I 4500
    if (TPS >=4){
        InjectionTime = rpm12[0];
    }
    else if(TPS >=3.85){
        InjectionTime = rpm12[1];
    }
    else if(TPS >=3.8){
        InjectionTime = rpm12[2];
    }
    else if(TPS >=3.75){
        InjectionTime = rpm12[3];
    }
    else if(TPS >=3.7){
        InjectionTime = rpm12[4];
    }
    else if(TPS >=3.65){
        InjectionTime = rpm12[5];
    }
    else if(TPS >=3.6){

InjectionTime = rpm12[6];
}
else if(TPS >=3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >=3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >=3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >=3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >=3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >=3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >=3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >=3.2){
    InjectionTime = rpm12[14];
}
else if(TPS >=3.15){
    InjectionTime = rpm12[15];
else if(TPS >= 3.1) {
    InjectionTime = rpm12[16];
}
else if(TPS >= 3.05) {
    InjectionTime = rpm12[17];
}
else if(TPS >= 3) {
    InjectionTime = rpm12[18];
}
else if(TPS >= 2.9) {
    InjectionTime = rpm12[19];
}
else if(TPS >= 2.8) {
    InjectionTime = rpm12[20];
}
else if(TPS >= 2.7) {
    InjectionTime = rpm12[21];
}
else if(TPS >= 2.6) {
    InjectionTime = rpm12[22];
}
else if(TPS >= 2.5) {
    InjectionTime = rpm12[23];
}
else if(TPS >= 2.4) {
    InjectionTime = rpm12[24];
}
else if(TPS >= 2.3) {
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2) {
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1) {
    InjectionTime = rpm12[27];
}
else if(TPS >= 2) {
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9) {
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8) {
    InjectionTime = rpm12[30];
}
else if(TPS >= 1.7) {
    InjectionTime = rpm12[31];
}
else if(TPS >= 1.6) {
    InjectionTime = rpm12[32];
}
else if(TPS >= 1.5) {
    InjectionTime = rpm12[33];
}
else if(TPS >= 1.4) {
InjectionTime = rpm12[34];
}
else if(TPS >=1.3){
    InjectionTime = rpm12[35];
}
else if(TPS >=1.2){
    InjectionTime = rpm12[36];
}
else if(TPS >=1.1){
    InjectionTime = rpm12[37];
}
else if(TPS >=1){
    InjectionTime = rpm12[38];
}
else if(TPS >=0.9){
    InjectionTime = rpm12[39];
}
else if(TPS >=0.8){
    InjectionTime = rpm12[40];
}

if ( 4500 < rpm && rpm < 4860 ) { 
    if (TPS >=4){
        InjectionTime = rpm12[0];
    }
    else if(TPS >=3.85){
        InjectionTime = rpm12[1];
    }
else if(TPS >= 3.8) {
    InjectionTime = rpm12[2];
}
else if(TPS >= 3.75) {
    InjectionTime = rpm12[3];
}
else if(TPS >= 3.7) {
    InjectionTime = rpm12[4];
}
else if(TPS >= 3.65) {
    InjectionTime = rpm12[5];
}
else if(TPS >= 3.6) {
    InjectionTime = rpm12[6];
}
else if(TPS >= 3.55) {
    InjectionTime = rpm12[7];
}
else if(TPS >= 3.5) {
    InjectionTime = rpm12[8];
}
else if(TPS >= 3.45) {
    InjectionTime = rpm12[9];
}
else if(TPS >= 3.4) {
    InjectionTime = rpm12[10];
}
else if(TPS >= 3.35) {

InjectionTime = rpm12[11];
}
else if(TPS >=3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >=3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >=3.2){
    InjectionTime = rpm12[14];
}
else if(TPS >=3.15){
    InjectionTime = rpm12[15];
}
else if(TPS >=3.1){
    InjectionTime = rpm12[16];
}
else if(TPS >=3.05){
    InjectionTime = rpm12[17];
}
else if(TPS >=3){
    InjectionTime = rpm12[18];
}
else if(TPS >=2.9){
    InjectionTime = rpm12[19];
}
else if(TPS >=2.8){
    InjectionTime = rpm12[20];
else if(TPS >= 2.7)
    {InjectionTime = rpm12[21];
    }
else if(TPS >= 2.6)
    {InjectionTime = rpm12[22];
    }
else if(TPS >= 2.5)
    {InjectionTime = rpm12[23];
    }
else if(TPS >= 2.4)
    {InjectionTime = rpm12[24];
    }
else if(TPS >= 2.3)
    {InjectionTime = rpm12[25];
    }
else if(TPS >= 2.2)
    {InjectionTime = rpm12[26];
    }
else if(TPS >= 2.1)
    {InjectionTime = rpm12[27];
    }
else if(TPS >= 2)
    {InjectionTime = rpm12[28];
    }
else if(TPS >= 1.9)
    {InjectionTime = rpm12[29];
    }
else if(TPS >=1.8){
    InjectionTime = rpm12[30];
}
else if(TPS >=1.7){
    InjectionTime = rpm12[31];
}
else if(TPS >=1.6){
    InjectionTime = rpm12[32];
}
else if(TPS >=1.5){
    InjectionTime = rpm12[33];
}
else if(TPS >=1.4){
    InjectionTime = rpm12[34];
}
else if(TPS >=1.3){
    InjectionTime = rpm12[35];
}
else if(TPS >=1.2){
    InjectionTime = rpm12[36];
}
else if(TPS >=1.1){
    InjectionTime = rpm12[37];
}
else if(TPS >=1){
    InjectionTime = rpm12[38];
}
else if(TPS >=0.9){
InjectionTime = rpm12[39];
}
else if(TPS >=0.8){
    InjectionTime = rpm12[40];
}
}

if ( 4860 < rpm && rpm < 5220) {  //PER REVOLUCIONS ENTRE 4860 I 5220
    if (TPS >=4){
        InjectionTime = rpm12[0];
    }
    else if(TPS >=3.85){
        InjectionTime = rpm12[1];
    }
    else if(TPS >=3.8){
        InjectionTime = rpm12[2];
    }
    else if(TPS >=3.75){
        InjectionTime = rpm12[3];
    }
    else if(TPS >=3.7){
        InjectionTime = rpm12[4];
    }
    else if(TPS >=3.65){
        InjectionTime = rpm12[5];
    }
    else if(TPS >=3.6){
        InjectionTime = rpm12[6];
    }
}
else if(TPS >=3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >=3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >=3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >=3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >=3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >=3.3){
    InjectionTime = rpm12[12];
}
else if(TPS >=3.25){
    InjectionTime = rpm12[13];
}
else if(TPS >=3.2){
    InjectionTime = rpm12[14];
}
else if(TPS >=3.15){
    InjectionTime = rpm12[15];
}
else if(TPS >=3.1){
InjectionTime = rpm12[16];
}
else if(TPS >=3.05){
    InjectionTime = rpm12[17];
}
else if(TPS >=3){
    InjectionTime = rpm12[18];
}
else if(TPS >=2.9){
    InjectionTime = rpm12[19];
}
else if(TPS >=2.8){
    InjectionTime = rpm12[20];
}
else if(TPS >=2.7){
    InjectionTime = rpm12[21];
}
else if(TPS >=2.6){
    InjectionTime = rpm12[22];
}
else if(TPS >=2.5){
    InjectionTime = rpm12[23];
}
else if(TPS >=2.4){
    InjectionTime = rpm12[24];
}
else if(TPS >=2.3){
    InjectionTime = rpm12[25];

else if(TPS >=2.2){
    InjectionTime = rpm12[26];
}
else if(TPS >=2.1){
    InjectionTime = rpm12[27];
}
else if(TPS >=2){
    InjectionTime = rpm12[28];
}
else if(TPS >=1.9){
    InjectionTime = rpm12[29];
}
else if(TPS >=1.8){
    InjectionTime = rpm12[30];
}
else if(TPS >=1.7){
    InjectionTime = rpm12[31];
}
else if(TPS >=1.6){
    InjectionTime = rpm12[32];
}
else if(TPS >=1.5){
    InjectionTime = rpm12[33];
}
else if(TPS >=1.4){
    InjectionTime = rpm12[34];
}
else if(TPS >= 1.3) {
    InjectionTime = rpm12[35];
}
else if(TPS >= 1.2) {
    InjectionTime = rpm12[36];
}
else if(TPS >= 1.1) {
    InjectionTime = rpm12[37];
}
else if(TPS >= 1) {
    InjectionTime = rpm12[38];
}
else if(TPS >= 0.9) {
    InjectionTime = rpm12[39];
}
else if(TPS >= 0.8) {
    InjectionTime = rpm12[40];
}
}

if (rpm > 5220) {
    if (TPS >= 4) {
        InjectionTime = rpm12[0];
    }
    else if(TPS >= 3.85) {
        InjectionTime = rpm12[1];
    }
    else if(TPS >= 3.8) {
        InjectionTime = rpm12[2];
    }
else if(TPS >= 3.75){
    InjectionTime = rpm12[3];
}
else if(TPS >= 3.7){
    InjectionTime = rpm12[4];
}
else if(TPS >= 3.65){
    InjectionTime = rpm12[5];
}
else if(TPS >= 3.6){
    InjectionTime = rpm12[6];
}
else if(TPS >= 3.55){
    InjectionTime = rpm12[7];
}
else if(TPS >= 3.5){
    InjectionTime = rpm12[8];
}
else if(TPS >= 3.45){
    InjectionTime = rpm12[9];
}
else if(TPS >= 3.4){
    InjectionTime = rpm12[10];
}
else if(TPS >= 3.35){
    InjectionTime = rpm12[11];
}
else if(TPS >= 3.3) {
    InjectionTime = rpm12[12];
}
else if(TPS >= 3.25) {
    InjectionTime = rpm12[13];
}
else if(TPS >= 3.2) {
    InjectionTime = rpm12[14];
}
else if(TPS >= 3.15) {
    InjectionTime = rpm12[15];
}
else if(TPS >= 3.1) {
    InjectionTime = rpm12[16];
}
else if(TPS >= 3.05) {
    InjectionTime = rpm12[17];
}
else if(TPS >= 3) {
    InjectionTime = rpm12[18];
}
else if(TPS >= 2.9) {
    InjectionTime = rpm12[19];
}
else if(TPS >= 2.8) {
    InjectionTime = rpm12[20];
}
else if(TPS >= 2.7) {

InjectionTime = rpm12[21];
}
else if(TPS >= 2.6){
    InjectionTime = rpm12[22];
}
else if(TPS >= 2.5){
    InjectionTime = rpm12[23];
}
else if(TPS >= 2.4){
    InjectionTime = rpm12[24];
}
else if(TPS >= 2.3){
    InjectionTime = rpm12[25];
}
else if(TPS >= 2.2){
    InjectionTime = rpm12[26];
}
else if(TPS >= 2.1){
    InjectionTime = rpm12[27];
}
else if(TPS >= 2){
    InjectionTime = rpm12[28];
}
else if(TPS >= 1.9){
    InjectionTime = rpm12[29];
}
else if(TPS >= 1.8){
    InjectionTime = rpm12[30];
} 
else if(TPS >= 1.7) {
    InjectionTime = rpm12[31];
} 
else if(TPS >= 1.6) {
    InjectionTime = rpm12[32];
} 
else if(TPS >= 1.5) {
    InjectionTime = rpm12[33];
} 
else if(TPS >= 1.4) {
    InjectionTime = rpm12[34];
} 
else if(TPS >= 1.3) {
    InjectionTime = rpm12[35];
} 
else if(TPS >= 1.2) {
    InjectionTime = rpm12[36];
} 
else if(TPS >= 1.1) {
    InjectionTime = rpm12[37];
} 
else if(TPS >= 1) {
    InjectionTime = rpm12[38];
} 
else if(TPS >= 0.9) {
    InjectionTime = rpm12[39];
}
else if (TPS >= 0.8) {
    InjectionTime = rpm12[40];
}
}
}

else{
    TIMSK3=0x00; // enabled global and timer overflow interrupt;
    TIMSK4=0x00; // enabled global and timer overflow interrupt;
    TIMSK5=0x00;
    digitalWrite(8,LOW); // Posem a nivell baix senyal d’encesa
    ValueAnalogPosition = analogRead(AnalogPosition); // Llegim TPS
    TPS = ((ValueAnalogPosition * 4.9)/1023); // Convertim valor TPS a analògic
}
break;
}

//INTERRUPCIONS
void RPM ()                       // cridem la interrupció RPM per flanc de pujada
{
    pulses++;                       // incrementem variable per cada pols CONTA REVOLUCIONS
    TCNT3=IgnitionLoad;            // Carreguem temps encesa. Funciona directament amb el valor, s’ha de fer amb variable
    TIMSK1=0x01;
    TIMSK3=0x01;
    PulseCounter++;              // Variable per injectar cada dos polsos del sensor cigüenyal

    if (PulseCounter == 1){

    

}
digitalWrite(22,HIGH);
digitalWrite(24,HIGH);
digitalWrite(26,HIGH);
digitalWrite(28,HIGH);
TCNT5=InjectionTime;
TIMSK5=0x01;
}
else{
PulseCounter = 0;
}

ISR(TIMER1_OVF_vect) {  // InterrupcióTimer per contar revolucions
TCNT1=0xE796;  // revolucions del cotxe
rpm = (pulses * 300);  // Fem càlcul per les revolucions. Cada 4 polsos és una
                       // revolució
pulses = 0;
}
ISR(TIMER3_OVF_vect) {  // InterrupcióTimer per temps d'encesa
digitalWrite(8,LOW);
TCNT4=0xD8F0;  // 16bit counter register
}

//INTEGRACIONS TEMPORITZADORS
ISR(TIMER4_OVF_vect) {  // InterrupcióTimer per temps d'encesa
digitalWrite(8,HIGH);
}
ISR(TIMER5_OVF_vect) {  // InterrupcióTimer per temps INJECCIO
digitalWrite(22,LOW);  // Funcionem injecció simultània. Activem els 4 injectors a la vegada.

digitalWrite(24,LOW);
digitalWrite(26,LOW);
digitalWrite(28,LOW);
}

2. Codi del programa Transferència

```c
#include <utility.h>    // LLIBRERIES LABWINDOWS
#include <formatio.h>
#include <ansi_c.h>
#include <rs232.h>
#include <cvirte.h>
#include <userint.h>
#include "ECU_CASU.h"

static int panel3, panel2, panel1;  // VARIABLES GLOBALS
char c[10], nbytes;
int i, comport=14, RS232Error, vel=19200, paritat=1, databits=8, stopbits=1, dadaenviar;
unsigned int inputq=256, outputq=256;

int main (int argc, char *argv[]) {
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1; /* out of memory */
    if ((panel3 = LoadPanel (0, "ECU_CASU.uir", PANEL_3)) < 0)
        return -1;
    if ((panel2 = LoadPanel (0, "ECU_CASU.uir", PANEL_2)) < 0)
        return -1;
    if ((panel1 = LoadPanel (0, "ECU_CASU.uir", PANEL_1)) < 0)
        return -1;

    DisplayPanel (panel1);
    RunUserInterface ();
    DiscardPanel (panel1);
    DiscardPanel (panel3);
    return 0;
}

int CVICALLBACK EXITBUTTON (int panel, int control, int event, void *callbackData, int eventData1, int eventData2) {
    switch (event) {
    case EVENT_COMMIT:
        QuitUserInterface (0);
        break;
    }
    return 0;
}
```
int CVICALLBACK CONFIGUREBUTTON (int panel, int control, int event, 
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
    case EVENT_COMMIT:
        HidePanel (panel1);
        DisplayPanel (panel2);
        break;
    }
    return 0;
}

int CVICALLBACK SENDBUTTON (int panel, int control, int event, 
    void *callbackData, int eventData1, int eventData2)
{
    float tempsinj, val; // VARIABLES LOCALS
    char cadena[6],cad[4], buffer[11], inici[1],final[1],n1;
    unsigned int val1, cont=0, col=1, fil=1, ind=0, rebut, transf=0, errtrans=0;

    switch (event)
    {
    case EVENT_COMMIT:
        ind=0;

        for (fil=1; fil<20;fil++) {
            for (col=1; col<42;col++) {
                GetTableCellVal(panel1, PANEL_1_TABLE, 
                                    MakePoint (col, fil), &val);
                val1 = val * (65535/32.768); // CONVERSIO VALOR
                val1 = 65535 - val1;
                SetCtrlVal(panel1,PANEL_1_TEMPINJ_2,val1);

                transf++; // Visualitzador grafic transferencia

                SetCtrlVal(panel1,PANEL_1_NUMERICALSLIDE,transf);

                Fmt (cad, "%s<%i[w3p0]", ind);
                // Posem nombre de 0 que falten per omplir. Resultat: "001"
                Fmt (cadena, "%s<%i[w5p0]", val1);
                // Posem nombre de 0 que falten per omplir. Resultat: "00001"

                inici[0]=0x41;         // Byte inici trama 'A'

                115
final[0]=0x47; // Byte final trama 'F'
buffer[0]=inici[0];
buffer[1]=cad[0];
buffer[2]=cad[1];
buffer[3]=cad[2];
buffer[4]=cadena[0];
buffer[5]=cadena[1];
buffer[6]=cadena[2];
buffer[7]=cadena[3];
buffer[8]=cadena[4];
buffer[9]=final[0];

ind++;
SetCtrlVal(panel1,PANEL_1_DADAENVIAR,buffer);
ComWrt (comport,buffer,10);

//ACKNOWLEDGE--------------------------------------------------------------
rebut=0;
errtrans;

while (rebut==0){
    n1=ComRdByte (comport); //COMPORT
    if (n1==0x59){
        // Si rebem una 'R'=0x59, correcte i sortim del while.
        SetCtrlVal(panel1,PANEL_1_DADAENVIAR_2,"Correcte");
        SetCtrlVal(panel1,PANEL_1_NUMERIC,ind-1);
        rebut++;
    }
    else {
        // Si rebem una 'N'=0x55, tornem a enviar dada.
        ComWrt (comport,buffer,10);
        errtrans++;
        if (errtrans>10){
            rebut++;
        }
    }
}
if (errtrans>10){
    fil=100; // Per sortir del FOR
    col=100;
    HidePanel (panel1);
    DisplayPanel (panel3);
int CVICALLBACK SAVEBUTTON (int panel, int control, int event, 
            void *callbackData, int eventData1, int eventData2) 
{ 
    switch (event) 
    { 
    case EVENT_COMMIT: 
        GetCtrlIndex (panel2, PANEL_2_COMPORT, &i); 
        GetLabelFromIndex (panel2, PANEL_2_COMPORT, i, c); 
        GetCtrlVal (panel2, PANEL_2_COMPORT, &comport); 
        GetCtrlVal (panel2, PANEL_2_BITSRATE, &vel); 
        GetCtrlVal (panel2, PANEL_2_PARITAT, &paritat); 
        GetCtrlVal (panel2, PANEL_2_BITSDADES, &databits); 
        GetCtrlVal (panel2, PANEL_2_BITSTOP, &stopbits); 
        GetCtrlVal (panel2, PANEL_2_CUAENTRADA, &inputq); 
        GetCtrlVal (panel2, PANEL_2_CUASORTIDA, &outputq); 
        RS232Error = OpenComConfig (comport, c, vel, paritat, databits, 
                    stopbits, inputq, outputq); 
        HidePanel (panel2); 
        DisplayPanel (panel1); 
        break; 
    } 
    return 0; 
} 

int CVICALLBACK DefaulButton (int panel, int control, int event, 
            void *callbackData, int eventData1, int eventData2) 
{ 
    float t=0.0; // Variables locals 
    int col=1; 
    int fil=1; 

    switch (event) 
    { 
    case EVENT_COMMIT: 
        col=1; 
        fil=1; 
        for (fil=1; fil<20;fil++) 
        { 

for (col=1; col<42; col++) {
    SetTableCellVal (panel1, PANEL_1_TABLE, MakePoint (col, fil), t);
}
break;
}
return 0;

int CVICALLBACK LoadButton (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    float t=0.0;  // Variables locals
    int col=1;
    int fil=1;

    switch (event)
    {
    case EVENT_COMMIT:
        col=1;
        fil=1;
        for (fil=1; fil<20; fil++) {
            for (col=1; col<42; col++) {
                if(t>=26.0){
                    t=26.0;
                }
                else {
                    t=t+0.1;
                }
            }
        }
        SetTableCellVal (panel1, PANEL_1_TABLE, MakePoint (col, fil), t);
    }
    break;
    }
    return 0;
}

int CVICALLBACK CopyClipboardButton (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    int err;  // Variables locals

    switch (event)
    {
    case EVENT_COMMIT:
        
    }
err = ClipboardPutTableVals (panel1, PANEL_1_TABLE, VAL_TABLE_ENTIRE_RANGE/*MakeRect(0,0,9,9)*/);

    break;
  }
  return 0;
}

int CVICALLBACK PasteClipboardButton (int panel, int control, int event, 
    void *callbackData, int eventData1, int eventData2)
{
    int err;    // Variables locals
    int available;

    switch (event)
    {
    case EVENT_COMMIT:

        err = ClipboardGetTableVals (panel1, PANEL_1_TABLE, VAL_TABLE_ENTIRE_RANGE, &available);

        break;
    }
    return 0;
}

int CVICALLBACK FAILBUTTON (int panel, int control, int event, 
    void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
    case EVENT_COMMIT:

        HidePanel (panel3);
        DisplayPanel (panel1);

        break;
    }
    return 0;
}
3. Esquemes de connexió

3.1 Esquema de connexió
3.2 Diseño encapsulado
### 4. Taules injecció

#### 4.1 Taules injecció Valors Standards

**POSICIÓ REGISTRE GUARDATS A LA EEPROM (REGISTRE 2000 fins 3557)**

<table>
<thead>
<tr>
<th>Valor 1</th>
<th>Valor 2</th>
<th>Valor 3</th>
<th>Valor 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VALORS GUARDATS A LA EEPROM (REGISTRE 2000 fins 3557)**

<table>
<thead>
<tr>
<th>Valor 1</th>
<th>Valor 2</th>
<th>Valor 3</th>
<th>Valor 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VALORS TAULA STANDARD LABWINDOWS PER DEFECTE (en ms)**

<table>
<thead>
<tr>
<th>Valor 1</th>
<th>Valor 2</th>
<th>Valor 3</th>
<th>Valor 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 4.2 Taules injecció Valors Modificats

#### POSICIO REGISTRE GUARDATS A LA EEPROM (REGISTRE 0 fins 1557)

<table>
<thead>
<tr>
<th>RPM</th>
<th>Valors</th>
<th>Modificats</th>
</tr>
</thead>
<tbody>
<tr>
<td>5200</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>4500</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>4000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>3200</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>2700</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>2000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>1000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
</tbody>
</table>

#### VALORS GUARDATS A LA EEPROM (REGISTRE 0 fins 1557)

<table>
<thead>
<tr>
<th>RPM</th>
<th>Valors</th>
<th>Modificats</th>
</tr>
</thead>
<tbody>
<tr>
<td>5200</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>4500</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>4000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>3200</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>2700</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>2000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>1000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
</tbody>
</table>

#### VALORS TAULA STANDARD LABWINDOWS PER DEFECTE (en ms)

<table>
<thead>
<tr>
<th>RPM</th>
<th>Valors</th>
<th>Modificats</th>
</tr>
</thead>
<tbody>
<tr>
<td>5200</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>4500</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>4000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>3200</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>2700</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>2000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
<tr>
<td>1000</td>
<td>0,000</td>
<td>0,000</td>
</tr>
</tbody>
</table>

---

122
5. Fitxa dels components

5.1 Transistor BD245C

<table>
<thead>
<tr>
<th>RATING</th>
<th>SYMBOL</th>
<th>VALUE</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collector-emitter voltage ($V_{CE}$ = 100 Ω)</td>
<td>$V_{CE}$</td>
<td>65</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>$V_{BO}$</td>
<td>60</td>
<td>V</td>
</tr>
<tr>
<td>Collector-emitter voltage ($I_{C}$ = 30 mA)</td>
<td>$V_{BE}$</td>
<td>45</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td>$I_{BO}$</td>
<td>60</td>
<td>A</td>
</tr>
<tr>
<td>Emitter-base voltage</td>
<td>$V_{BO}$</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>Continuous collector current</td>
<td>$I_{C}$</td>
<td>15</td>
<td>A</td>
</tr>
<tr>
<td>Peak collector current (see Note 1)</td>
<td>$I_{CM}$</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>Continuous base current</td>
<td>$I_{B}$</td>
<td>60</td>
<td>W</td>
</tr>
<tr>
<td>Continuous device dissipation at (or below) 25°C case temperature (see Note 2)</td>
<td>$P_{D}$</td>
<td>3</td>
<td>W</td>
</tr>
<tr>
<td>Unclamped inductive load-energy (see Note 4)</td>
<td>$W_{L,CL}$</td>
<td>62.5</td>
<td>mJ</td>
</tr>
<tr>
<td>Operating junction temperature range</td>
<td>$T_{J}$</td>
<td>-65 to +150</td>
<td>°C</td>
</tr>
<tr>
<td>Storage temperature range</td>
<td>$T_{ST}$</td>
<td>250</td>
<td>°C</td>
</tr>
<tr>
<td>Load temperature 3.2 mm from case for 10 seconds</td>
<td>$T_{L}$</td>
<td>125</td>
<td>°C</td>
</tr>
</tbody>
</table>

NOTES:
1. This value applies for $t_{1}$ ≤ 0.3 ms, duty cycle ≤ 10%.
2. Derate linearly to 150°C case temperature at the rate of 0.54 W/°C.
3. Derate linearly to 150°C free air temperature at the rate of 24 W/°C.
4. This rating is based on the capability of the transistor to operate safely in a circuit of: $L = 20$ mm, $I_{CM} = 0.4$ A, $R_{DC} = 100$ Ω, $V_{CE}$ = 0 V, $R_{D}$ = 0.1 Ω, $V_{CC}$ = 20 V.
**BD245, BD245A, BD245B, BD245C**

**NPN SILICON POWER TRANSISTORS**

**JUNE 1973 - REVISED MARCH 1997**

**electrical characteristics at 25°C case temperature**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>BD245</th>
<th>BD245A</th>
<th>BD245B</th>
<th>BD245C</th>
</tr>
</thead>
<tbody>
<tr>
<td>(V_{CEO}) Collector-emitter</td>
<td>(I_C = 30) mA (I_E = 0)</td>
<td>45</td>
<td>60</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>breakdown voltage</td>
<td></td>
<td></td>
<td>BD245A</td>
<td>BD245B</td>
<td>BD245C</td>
</tr>
<tr>
<td>(I_{CEO}) Collector-emitter cut-off</td>
<td>(V_{CE} = 55) V (V_{BE} = 0)</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>current</td>
<td></td>
<td></td>
<td>BD245A</td>
<td>BD245B</td>
<td>BD245C</td>
</tr>
<tr>
<td>(I_{CEO}) Collector cut-off</td>
<td>(V_{CE} = 30) V (I_E = 0)</td>
<td>0.7</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>current</td>
<td></td>
<td></td>
<td>BD245A</td>
<td>BD245B</td>
<td>BD245C</td>
</tr>
<tr>
<td>(I_{CEO}) Emitter cut-off current</td>
<td>(V_{CE} = 5) V (I_C = 0)</td>
<td>1</td>
<td></td>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BD245A</td>
<td>BD245B</td>
<td>BD245C</td>
</tr>
<tr>
<td>(h_{FE}) Forward current transfer</td>
<td>(V_{CE} = 4) V (I_C = 1)</td>
<td>40</td>
<td>20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ratio</td>
<td></td>
<td></td>
<td>BD245A</td>
<td>BD245B</td>
<td>BD245C</td>
</tr>
<tr>
<td>(V_{CEO\text{sat}}) Collector-emitter</td>
<td>(I_E = 0.3) A (I_C = 3)</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
</tr>
<tr>
<td>saturation voltage</td>
<td></td>
<td></td>
<td>BD245A</td>
<td>BD245B</td>
<td>BD245C</td>
</tr>
<tr>
<td>(V_{CEO\text{sat}}) Collector-emitter</td>
<td>(I_E = 2.5) A (I_C = 10)</td>
<td></td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>saturation voltage</td>
<td></td>
<td></td>
<td>BD245A</td>
<td>BD245B</td>
<td>BD245C</td>
</tr>
<tr>
<td>(I_{CEO\text{sat}}) Collector-emitter</td>
<td>(I_E = 10) V (I_C = 0.5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>saturation transfer ratio</td>
<td></td>
<td></td>
<td>BD245A</td>
<td>BD245B</td>
<td>BD245C</td>
</tr>
<tr>
<td>(f_{CEO\text{sat}}) Collector-emitter</td>
<td>(I_E = 10) V (I_C = 0.5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>saturation transfer ratio</td>
<td></td>
<td></td>
<td>BD245A</td>
<td>BD245B</td>
<td>BD245C</td>
</tr>
</tbody>
</table>

**NOTES:**
5. These parameters must be measured using pulse techniques, \(t_p = 300\) \(\mu\)s, duty cycle \(\leq 2\%\).
6. These parameters must be measured using voltage-sensing contacts, separate from the current carrying contacts.

**thermal characteristics**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_{J\text{SC}}) Junction to case thermal resistance</td>
<td>1.56</td>
<td></td>
<td></td>
<td>°C/W</td>
</tr>
<tr>
<td>(R_{J\text{SA}}) Junction to free air thermal resistance</td>
<td>42</td>
<td></td>
<td></td>
<td>°C/W</td>
</tr>
</tbody>
</table>

**resistive-load-switching characteristics at 25°C case temperature**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I_{ON}) Turn-on time</td>
<td>(V_{CES\text{on}} = -3.7) V (R_L = 20) ohms</td>
<td>0.3</td>
<td></td>
<td></td>
<td>µs</td>
</tr>
<tr>
<td>(I_{OFF}) Turn-off time</td>
<td>(V_{CES\text{off}} = 0) V (I_E = 0)</td>
<td>-1</td>
<td></td>
<td></td>
<td>µs</td>
</tr>
</tbody>
</table>

**PRODUCT INFORMATION**

124
MECHANICAL DATA

SOT-93
3-pin plastic flange-mount package
This single-in-line package consists of a circuit mounted on a lead frame and encapsulated within a plastic compound. The compound will withstand soldering temperature with no deformation, and circuit performance characteristics will remain stable when operated in high humidity conditions. Leads require no additional cleaning or processing when used in soldered assembly.

NOTE A: The centre pin is in electrical contact with the mounting tab.

ALL LINEAR DIMENSIONS IN MILLIMETERS

PRODUCT INFORMATION

Power INNOVATIONS
IMPORTANT NOTICE

Power Innovations Limited (PI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to verify, before placing orders, that the information being relied on is current.

PI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with PI's standard warranty. Testing and other quality control techniques are utilized to the extent PI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except as mandated by government requirements.

PI accepts no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor is any license, either express or implied, granted under any patent right, copyright, design right, or other intellectual property right of PI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

PI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS.

Copyright © 1997, Power Innovations Limited
This datasheet has been downloaded from:

www.datasheetcatalog.com

Datasheets for electronics components.
5.2 Transistor TIP142

TIP140/141/142
TIP145/146/147

COMPLEMENTARY SILICON POWER DARLINGTON TRANSISTORS

- TIP141, TIP142, TIP145 AND TIP147 ARE STMicroelectronics PREFERRED SALES TYPES
- COMPLEMENTARY PNP - NPN DEVICES
- MONOLITHIC DARLINGTON CONFIGURATION
- INTEGRATED ANTI-PARALLEL COLLECTOR-EMITTER DIODE

APPLICATIONS
- LINEAR AND SWITCHING INDUSTRIAL EQUIPMENT

DESCRIPTION
The TIP140, TIP141 and TIP142 are silicon Epi-Base NPN power transistors in monolithic Darlington configuration, mounted in TO-218 plastic package. They are intended for use in power linear and switching applications. The complementary PNP types are TIP145, TIP146 and TIP147 respectively.

ABSOLUTE MAXIMUM RATINGS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCEO</td>
<td>Collector-Base Voltage (Ie = 0)</td>
<td>50</td>
<td>V</td>
</tr>
<tr>
<td>VCEO</td>
<td>Collector-Emitter Voltage (Ie = 0)</td>
<td>50</td>
<td>V</td>
</tr>
<tr>
<td>VCEO</td>
<td>Emitter-Base Voltage (Ic = 0)</td>
<td>5</td>
<td>V</td>
</tr>
<tr>
<td>IC</td>
<td>Collector Current</td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td>IC(PS)</td>
<td>Collector Peak Current</td>
<td>20</td>
<td>A</td>
</tr>
<tr>
<td>IR</td>
<td>Base Current</td>
<td>0.5</td>
<td>A</td>
</tr>
<tr>
<td>PT</td>
<td>Total Dissipation at Tj = 60°C</td>
<td>105</td>
<td>W</td>
</tr>
<tr>
<td>Tj</td>
<td>Storage Temperature</td>
<td>-55 to 150</td>
<td>°C</td>
</tr>
<tr>
<td>Tj</td>
<td>Max. Operating Junction Temperature</td>
<td>150</td>
<td>°C</td>
</tr>
</tbody>
</table>

For PNP types, voltage and current values are negative.

March 2000
### THERMAL DATA

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
</table>
| Tjm   | Junction-to-Mounting Temperature | for TIP140/145: Vgs = 80 V  
for TIP141/146: Vgs = 80 V  
for TIP142/147: Vgs = 100 V | 1  | 1   | 1   | °C/W |

### ELECTRICAL CHARACTERISTICS (Tamb = 25 °C unless otherwise specified)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
</table>
| Ic(o)   | Collector Cut-off Current (Ic = 0) | for TIP140/145: Vgs = 60 V  
for TIP141/146: Vgs = 80 V  
for TIP142/147: Vgs = 100 V | 1  | 1   | 1   | mA   |
| Ic(o)   | Collector Cut-off Current (Ic = 0) | for TIP140/145: Vgs = 30 V  
for TIP141/146: Vgs = 40 V  
for TIP142/147: Vgs = 50 V | 2  | 2   | 2   | mA   |
| Ie(o)   | Emitter Cut-off Current (Ie = 0)  | Vgs = 5 V                | 2   |     |     | mA   |

- **Vce(on)**: Collector-Emitter Sustaining Voltage (Ic = 0)  
  - for TIP140/145: 60 V  
  - for TIP141/146: 90 V  
  - for TIP142/147: 100 V  
  | Min. | Typ. | Max. | Unit |
  | 50   | 90   | 100  | V    |

- **Vces(on)**: Collector-Emitter Saturation Voltage  
  - Ic = 5 A  
  - Ic = 10 A  
  | Ic | Typ. | Max. | Unit |
  | 10 A | 10 mA | 40 mA | V    |

- **Vbe(on)**: Base-Emitter Voltage  
  - Ic = 10 A  
  | Ic | Typ. | Max. | Unit |
  | 10 A | 4 V  | 4 V  | V    |

- **Ics** (DC Current Gain)  
  - Ic = 10 A  
  | Ic | Typ. | Max. | Unit |
  | 10 A | 40 mA | 40 mA | V    |

- **ttr** (Turn-on Time)  
  - Ic = 10 A  
  | Ic | Typ. | Max. | Unit |
  | 10 A | 40 mA | 40 mA | V    |

For PNP types, voltage and current values are negative.

*Pulse Test: Pulse duration = 300μs, duty cycle 1.5%
### TO-218 (SOT-83) MECHANICAL DATA

<table>
<thead>
<tr>
<th>DIM.</th>
<th>mm</th>
<th>INCH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIN.</td>
<td>TYP.</td>
</tr>
<tr>
<td>A</td>
<td>4.7</td>
<td>4.0</td>
</tr>
<tr>
<td>B</td>
<td>0.5</td>
<td>0.78</td>
</tr>
<tr>
<td>C</td>
<td>1.17</td>
<td>1.37</td>
</tr>
<tr>
<td>D</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>1.1</td>
<td>1.3</td>
</tr>
<tr>
<td>Q</td>
<td>10.8</td>
<td>11.1</td>
</tr>
<tr>
<td>H</td>
<td>14.7</td>
<td>15.2</td>
</tr>
<tr>
<td>L2</td>
<td>–</td>
<td>15.2</td>
</tr>
<tr>
<td>L3</td>
<td>–</td>
<td>18</td>
</tr>
<tr>
<td>L5</td>
<td>3.00</td>
<td>4.15</td>
</tr>
<tr>
<td>L6</td>
<td>3.1</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>–</td>
<td>12.2</td>
</tr>
<tr>
<td>Ø</td>
<td>4</td>
<td>4.1</td>
</tr>
</tbody>
</table>
5.3 Regulador LM7805

5.4 Regulador LM7809

MC78XX/LM78XX/MC78XXA
3-Terminal 1A Positive Voltage Regulator

Features
• Output Current up to 1A
• Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
• Thermal Overload Protection
• Short Circuit Protection
• Output Transistor Safe Operating Area Protection

Description
The MC78XX-LM78XX-MC78XXA series of three terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

Internal Block Diagram

Rev. 1.0.1

©2001 Fairchild Semiconductor Corporation
Absolute Maximum Ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Voltage (for $V_D = 5\text{V}$ to $18\text{V}$)</td>
<td>$V_I$</td>
<td>35</td>
<td>V</td>
</tr>
<tr>
<td>(for $V_D = 24\text{V}$)</td>
<td>$V_I$</td>
<td>40</td>
<td>V</td>
</tr>
<tr>
<td>Thermal Resistance Junction-Cases (TO-220)</td>
<td>$R_{JIC}$</td>
<td>5</td>
<td>°C/W</td>
</tr>
<tr>
<td>Thermal Resistance Junction-Air (TO-220)</td>
<td>$R_{JJA}$</td>
<td>65</td>
<td>°C/W</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>$T_{OPR}$</td>
<td>0°C to +125°C</td>
<td></td>
</tr>
<tr>
<td>Storage Temperature Range</td>
<td>$T_{STG}$</td>
<td>-65°C to +150°C</td>
<td></td>
</tr>
</tbody>
</table>

Electrical Characteristics (MC7805/LM7805)

(Refer to test circuit, $0°C < T_J < 125°C$, $I_O = 500\text{mA}$, $V_I = 10\text{V}$, $C_I = 0.33\mu\text{F}$, $C_O = 0.1\mu\text{F}$, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>MC7805/LM7805</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Unit</td>
</tr>
<tr>
<td>Output Voltage</td>
<td>$V_O$</td>
<td>$T_J = +25°C$</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5.0mA $\leq I_O \leq 1.0\text{A}$, $P_O \leq 15\text{W}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$V_I = 7\text{V}$ to $20\text{V}$</td>
</tr>
<tr>
<td>Line Regulation (Note 1)</td>
<td>$R_{oline}$</td>
<td>$T_J = +25°C$</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$V_O = 7\text{V}$ to $25\text{V}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$V_I = 8\text{V}$ to $12\text{V}$</td>
</tr>
<tr>
<td>Load Regulation (Note 1)</td>
<td>$R_{load}$</td>
<td>$T_J = +25°C$</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$I_O = 5.0\text{mA}$ to $1.5\text{A}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$I_O = 250\text{mA}$ to $750\text{mA}$</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>$I_Q$</td>
<td>$T_J = +25°C$</td>
<td>-</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>$\Delta I_Q$</td>
<td>$I_O = 5\text{mA}$ to $1.0\text{A}$</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 7\text{V}$ to $25\text{V}$</td>
<td>-</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>$\Delta V_O/\Delta T$</td>
<td>$I_O = 5\text{mA}$</td>
<td>-</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>$V_{N}$</td>
<td>$f = 10\text{Hz}$ to $100\text{KHz}$, $T_A = +25°C$</td>
<td>-</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>$RR$</td>
<td>$f = 120\text{Hz}$</td>
<td>62</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>$V_{Drop}$</td>
<td>$I_O = 1\text{A}$, $T_J = +25°C$</td>
<td>-</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>$r_O$</td>
<td>$f = 1\text{KHz}$</td>
<td>-</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>$I_{SC}$</td>
<td>$V_I = 35\text{V}$, $T_A = +25°C$</td>
<td>-</td>
</tr>
<tr>
<td>Peak Current</td>
<td>$I_{PK}$</td>
<td>$T_J = +25°C$</td>
<td>-</td>
</tr>
</tbody>
</table>

Note:
1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.
## Electrical Characteristics (MC7806)

(Refer to test circuit: \(0^\circ C < T_J < 125^\circ C\), \(I_O = 500mA\), \(V_I = 11V\), \(C_I = 0.33\mu F\), \(C_O = 0.1\mu F\), unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>MC7806</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>(V_O)</td>
<td>(T_J = +25^\circ C) (5.0mA \leq I_O \leq 1.0A), (P_O \leq 15W) (V_I = 8.0V) to (21V)</td>
<td>Min. 5.7</td>
<td>Typ. 6.0</td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>(R_{line})</td>
<td>(T_J = +25^\circ C) (V_I = 8V) to (25V) (V_I = 9V) to (13V)</td>
<td>- 5</td>
<td>120 mV</td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>(R_{load})</td>
<td>(T_J = +25^\circ C) (I_O = 5mA) to (1.5A) (I_O = 250mA) to (50A)</td>
<td>- 9</td>
<td>120 mV</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>(I_Q)</td>
<td>(T_J = +25^\circ C) (V_I = 8V) to (25V) (I_O = 5mA) to (1A)</td>
<td>- 5.0</td>
<td>8.0 mA</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>(\Delta I_Q)</td>
<td>(V_I = 8V) to (25V) (I_O = 5mA) to (1A) (I_O = 250mA) to (50A)</td>
<td>- 0.5</td>
<td>-</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>(\Delta V_O/\Delta T)</td>
<td>(I_O = 5mA) (f = 10Hz) to (100KHz) (T_A = +25^\circ C)</td>
<td>- 45</td>
<td>- (mV/\degree C)</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>(V_N)</td>
<td>(f = 120Hz) (V_I = 9V) to (19V) (f = 1kHz)</td>
<td>- 59</td>
<td>75 dB</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>(R_{ripple})</td>
<td>(f = 120Hz) (V_I = 9V) to (19V) (f = 1kHz)</td>
<td>- 2</td>
<td>- V</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>(V_{drop})</td>
<td>(I_O = 1A) (T_J = +25^\circ C) (f = 1kHz)</td>
<td>- 19</td>
<td>- (m\Omega)</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>(R_O)</td>
<td>(f = 1kHz) (V_I = 35V) (T_A = +25^\circ C)</td>
<td>- 250</td>
<td>- mA</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>(I_{SC})</td>
<td>(V_I = 35V) (T_A = +25^\circ C) (f = 1kHz)</td>
<td>- 2.2</td>
<td>- A</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Changes in \(V_O\) due to heating effects must be taken into account separately. Pulse testing at low duty is used.

---

137
## Electrical Characteristics (MC7808)

(Refer to test circuit. 0°C < T<sub>j</sub> < 125°C, I<sub>0</sub> = 500mA, V<sub>i</sub> =14V, C<sub>i</sub>= 0.33μF, C<sub>0</sub>= 0.1μF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>MC7808</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min.</td>
</tr>
<tr>
<td>Output Voltage</td>
<td>V&lt;sub&gt;0&lt;/sub&gt;</td>
<td>T&lt;sub&gt;j&lt;/sub&gt; =+25°C 5.0mA ≤ I&lt;sub&gt;0&lt;/sub&gt; ≤ 1.0A, P&lt;sub&gt;0&lt;/sub&gt; ≤ 15W</td>
<td>7.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V&lt;sub&gt;i&lt;/sub&gt; = 10.6V to 23V</td>
<td>6.0</td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Regline</td>
<td>T&lt;sub&gt;j&lt;/sub&gt; =+25°C V&lt;sub&gt;i&lt;/sub&gt; = 10.5V to 25V</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>160</td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>T&lt;sub&gt;j&lt;/sub&gt; =+25°C I&lt;sub&gt;0&lt;/sub&gt; = 5.0mA to 1.5A</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>160</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I&lt;sub&gt;0&lt;/sub&gt; = 250mA to 750mA</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>I&lt;sub&gt;0&lt;/sub&gt;</td>
<td>T&lt;sub&gt;j&lt;/sub&gt; =+25°C</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>ΔI&lt;sub&gt;0&lt;/sub&gt;</td>
<td>I&lt;sub&gt;0&lt;/sub&gt; = 5mA to 1.0A</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V&lt;sub&gt;i&lt;/sub&gt; = 10.6A to 25V</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>ΔV&lt;sub&gt;0&lt;/sub&gt;/ΔT</td>
<td>I&lt;sub&gt;0&lt;/sub&gt; = 5mA</td>
<td>-0.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>V&lt;sub&gt;N&lt;/sub&gt;</td>
<td>f = 10Hz to 100kHz, T&lt;sub&gt;A&lt;/sub&gt; =+25°C</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-μV/Vo</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz, V&lt;sub&gt;i&lt;/sub&gt; = 11.5V to 21.5V</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>73</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>V&lt;sub&gt;Drop&lt;/sub&gt;</td>
<td>I&lt;sub&gt;0&lt;/sub&gt; = 1A, T&lt;sub&gt;j&lt;/sub&gt; =+25°C</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>ro</td>
<td>f = 1kHz</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-μΩ</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>I&lt;sub&gt;SC&lt;/sub&gt;</td>
<td>V&lt;sub&gt;i&lt;/sub&gt; = 35V, T&lt;sub&gt;A&lt;/sub&gt; =+25°C</td>
<td>230</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>IPK</td>
<td>T&lt;sub&gt;j&lt;/sub&gt; =+25°C</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Changes in V<sub>0</sub> due to heating effects must be taken into account separately. Pulse testing with low duty is used.
### Electrical Characteristics (MC7809)

(Refer to test circuit. $0^\circ C < T_J < 125^\circ C$, $I_O = 500mA$, $V_I = 15V$, $C_I = 0.33\mu F$, $C_O = 0.1\mu F$, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>MC7809</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min.</td>
<td>Typ.</td>
<td>Max.</td>
</tr>
<tr>
<td>Output Voltage</td>
<td>$V_O$</td>
<td>$T_J = +25^\circ C$</td>
<td>8.65</td>
<td>9</td>
<td>9.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$5.0mA$, $I_O \leq 1.0mA$, $P_O \leq 15W$</td>
<td>8.6</td>
<td>9</td>
<td>9.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 11.5V$ to $24V$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line Regulation (Notes1)</td>
<td>Regline</td>
<td>$T_J = +25^\circ C$</td>
<td>-</td>
<td>6</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 11.5V$ to $25V$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 12V$ to $17V$</td>
<td>-</td>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>$T_J = +25^\circ C$</td>
<td>-</td>
<td>12</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 5mA$ to $1.5A$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 250mA$ to $750mA$</td>
<td>-</td>
<td>4</td>
<td>90</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>$I_Q$</td>
<td>$T_J = +25^\circ C$</td>
<td>-</td>
<td>5.0</td>
<td>8.0</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>$\Delta I_Q$</td>
<td>$I_O = 5mA$ to $1.0A$</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 11.5V$ to $26V$</td>
<td>-</td>
<td>-</td>
<td>1.3</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>$\Delta V_O/\Delta T$</td>
<td>$I_O = 5mA$</td>
<td>-</td>
<td>-1</td>
<td>-</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>$V_N$</td>
<td>$f = 10Hz$ to $100KHz$, $T_A = +25^\circ C$</td>
<td>-</td>
<td>58</td>
<td>$\mu V/V_o$</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>$R_R$</td>
<td>$f = 120Hz$</td>
<td>56</td>
<td>71</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 13V$ to $23V$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>$V_{Drop}$</td>
<td>$I_O = 1A$, $T_J = +25^\circ C$</td>
<td>-</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>$R_O$</td>
<td>$f = 1KHz$</td>
<td>-</td>
<td>17</td>
<td>-</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>$I_{SC}$</td>
<td>$V_I = 35V$, $T_A = +25^\circ C$</td>
<td>-</td>
<td>250</td>
<td>-</td>
</tr>
<tr>
<td>Peak Current</td>
<td>$I_{pk}$</td>
<td>$T_J = +25^\circ C$</td>
<td>-</td>
<td>2.2</td>
<td>-</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.
## Electrical Characteristics (MC7810)

(Refer to test circuit. 0°C ⩽ T_J ⩽ 125°C, I_O = 500mA, V_IN = 16V, C = 0.33μF, C = 0.1μF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>V_O</td>
<td>T_J = ±25°C</td>
<td>5.0mA ⩽ I_O ⩽ 1.0A, P_O ⩽ 15W</td>
<td>9.6</td>
<td>10</td>
<td>10.4</td>
<td>V</td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Rgline</td>
<td>T_J = ±25°C</td>
<td>V_IN = 12.5V to 25V</td>
<td>-</td>
<td>10</td>
<td>200</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>V_IN = 13V to 25V</td>
<td>-</td>
<td>3</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Rload</td>
<td>T_J = ±25°C</td>
<td>I_O = 5mA to 1.5A</td>
<td>-</td>
<td>12</td>
<td>200</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I_O = 250mA to 750mA</td>
<td>-</td>
<td>4</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>I_Q</td>
<td>T_J = ±25°C</td>
<td>-</td>
<td>5.1</td>
<td>8.0</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>ΔI_Q</td>
<td>I_O = 5mA to 1.0A</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>V_IN = 12.5V to 25V</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>ΔV_O/ΔT</td>
<td>I_O = 5mA</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>mV/°C</td>
<td></td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>V_N</td>
<td>f = 10Hz to 100KHz, T_A = ±25°C</td>
<td>-</td>
<td>58</td>
<td>-</td>
<td>μV/V/V_O</td>
<td></td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz</td>
<td>V_IN = 13V to 23V</td>
<td>66</td>
<td>71</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>V_Drop</td>
<td>I_O = 1A, T_J = ±25°C</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Output Resistance</td>
<td>r_O</td>
<td>f = 1KHz</td>
<td>-</td>
<td>17</td>
<td>-</td>
<td>mΩ</td>
<td></td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>I_SC</td>
<td>V_IN = 35V, T_A = ±25°C</td>
<td>-</td>
<td>250</td>
<td>-</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Peak Current</td>
<td>I_PK</td>
<td>T_J = ±25°C</td>
<td>-</td>
<td>2.2</td>
<td>-</td>
<td>A</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty is used.
### Electrical Characteristics (MC7812)

(Refer to test circuit, $0^\circ{C} < T_J < 125^\circ{C}$, $I_O = 500\text{mA}$, $V_I = 15\text{V}$, $C = 0.33\mu\text{F}$, $C_O = 0.1\mu\text{F}$, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>MC7812</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min.</td>
</tr>
<tr>
<td>Output Voltage</td>
<td>$V_O$</td>
<td>$T_J = \pm 25^\circ{C}$; $I_O \leq 1.0\text{A}$; $P_O \leq 15\text{W}$; $V_I = 14.5\text{V}$ to $27\text{V}$</td>
<td>11.6</td>
</tr>
<tr>
<td>Line Regulation (Note 1)</td>
<td>$R_{LG}$</td>
<td>$T_J = \pm 25^\circ{C}$; $V_I = 14.5\text{V}$ to $30\text{V}$</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Load Regulation (Note 1)</td>
<td>$R_{LD}$</td>
<td>$T_J = \pm 25^\circ{C}$; $I_O = 5\text{mA}$ to $1.8\text{A}$</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>$I_Q$</td>
<td>$T_J = \pm 25^\circ{C}$; $V_I = 14.5\text{V}$ to $30\text{V}$</td>
<td>-</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>$\Delta I_Q$</td>
<td>$I_O = 5\text{mA}$ to $1.0\text{A}$</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>$\Delta V_O/\Delta T$</td>
<td>$I_O = 5\text{mA}$</td>
<td>-</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>$V_N$</td>
<td>$f = 10\text{Hz}$ to $100\text{kHz}$; $T_A = \pm 25^\circ{C}$</td>
<td>-</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>$RR$</td>
<td>$f = 120\text{Hz}$; $V_I = 15\text{V}$ to $25\text{V}$</td>
<td>55</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>$V_{DROP}$</td>
<td>$I_O = 1\text{A}$; $T_J = \pm 25^\circ{C}$</td>
<td>-</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>$R_O$</td>
<td>$f = 1\text{kHz}$</td>
<td>-</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>$I_{SC}$</td>
<td>$V_I = 35\text{V}$; $T_A = \pm 25^\circ{C}$</td>
<td>-</td>
</tr>
<tr>
<td>Peak Current</td>
<td>$I_{PK}$</td>
<td>$T_J = \pm 25^\circ{C}$</td>
<td>-</td>
</tr>
</tbody>
</table>

**Note:**

1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.
### Electrical Characteristics (MC7815)

(Refer to test circuit. 0°C < TJ < 125°C, IO = 500mA, V1 = 23V, C1 = 0.33μF, C2 = 0.1μF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>MC7815</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min.</td>
<td>Typ.</td>
</tr>
<tr>
<td>Output Voltage</td>
<td>VO</td>
<td>Tj = ±25°C</td>
<td>14.4</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.0mA &lt; IO ≤ 1.0A, P0 ≤ 15W, V1 = 17.5V to 30V</td>
<td>14.25</td>
<td>15</td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Regline</td>
<td>Tj = ±25°C</td>
<td>-11</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V1 = 17.5V to 30V</td>
<td>-3</td>
<td>150</td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>Tj = ±25°C</td>
<td>-12</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 8mA to 1.6A</td>
<td>-4</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 250mA to 7.5mA</td>
<td>-</td>
<td>150</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>IQ</td>
<td>Tj = ±25°C</td>
<td>-</td>
<td>5.2</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>ΔIQ</td>
<td>IO = 5mA to 1.0A</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V1 = 17.5V to 30V</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>ΔVO/ΔT</td>
<td>IO = 5mA</td>
<td>-1</td>
<td>-</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>VN</td>
<td>f = 10Hz to 100kHz, TA = ±25°C</td>
<td>-</td>
<td>90</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz, V1 = 18.5V to 28.5V</td>
<td>54</td>
<td>70</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>VDrop</td>
<td>IO = 1A, Tj = ±25°C</td>
<td>-2</td>
<td>-</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>RO</td>
<td>f = 1kHz</td>
<td>-19</td>
<td>-</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>ISC</td>
<td>V1 = 35V, Tj = ±25°C</td>
<td>-250</td>
<td>-</td>
</tr>
<tr>
<td>Peak Current</td>
<td>IPK</td>
<td>Tj = ±25°C</td>
<td>-2.2</td>
<td>-</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Changes in VO due to heating effects must be taken into account separately. Pulse testing with low duty is used.
### Electrical Characteristics (MC7818)

(Refer to test circuit, 0°C < T_J < 125°C, I_O = 500mA, V_I = 27V, C_I = 0.33μF, C_O = 0.1μF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>MC7818</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Min.</td>
</tr>
<tr>
<td>Output Voltage</td>
<td>VO</td>
<td>T_J = +25°C</td>
<td>17.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.0mA ≤ I_O ≤ 1.0A, P_O ≤ 15W</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>V_I = 21V to 33V</td>
<td>17.1</td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Regline</td>
<td>T_J = +25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>V_I = 21V to 33V</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V_I = 24V to 30V</td>
<td>-</td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>T_J = +25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>I_O = 1mA to 1.5A</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I_O = 250mA to 750mA</td>
<td>-</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>I_Q</td>
<td>T_J = +25°C</td>
<td>-</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>I_Q</td>
<td>V_I = 21V to 33V</td>
<td>-</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>V_OS</td>
<td>I_O = 5mA</td>
<td>-</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>V_N</td>
<td>f = 10kHz to 100kHz, T_A = +25°C</td>
<td>-</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz</td>
<td>V_I = 22V to 32V</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>V_Drop</td>
<td>I_O = 1A, T_J = +25°C</td>
<td>-</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>R_O</td>
<td>f = 1kHz</td>
<td>-</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>I_SC</td>
<td>V_I = 35V, T_A = +25°C</td>
<td>-</td>
</tr>
<tr>
<td>Peak Current</td>
<td>I_PK</td>
<td>T_J = +25°C</td>
<td>-</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty is used.
### Electrical Characteristics (MC7824)

(Refer to test circuit: $0^\circ C < T_J < 125^\circ C$, $I_O = 500mA$, $V_I = 33V$, $C = 0.33\mu F$, $C_O = 0.1\mu F$, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>$V_O$</td>
<td>$T_J = +25^\circ C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$5.0mA &lt; I_O &lt; 1.0A$, $P_O &lt; 15W$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 27V$ to $38V$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$22.8$ to $24.25V$</td>
</tr>
<tr>
<td>Line Regulation (Note 1)</td>
<td>$Regline$</td>
<td>$T_J = +25^\circ C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 27V$ to $38V$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-17$ to $480mV$</td>
</tr>
<tr>
<td>Load Regulation (Note 1)</td>
<td>$Reload$</td>
<td>$T_J = +25^\circ C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 5mA$ to $1.5A$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-15$ to $480mV$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 250mA$ to $750mA$</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>$I_Q$</td>
<td>$T_J = +25^\circ C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-5.2$ to $8.0mA$</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>$\Delta I_Q$</td>
<td>$V_I = 27V$ to $38V$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-0.1$ to $0.5mA$</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>$\Delta V_O/\Delta T$</td>
<td>$I_O = 5mA$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-1.5$ to $-mV/\circ C$</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>$V_N$</td>
<td>$f = 10Hz$ to $100kHz$, $T_A = +25^\circ C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-60$ to $-\mu V/V_o$</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>$RR$</td>
<td>$f = 120Hz$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 28V$ to $38V$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$50$ to $67\ dB$</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>$V_{Drop}$</td>
<td>$I_O = 1A$, $T_J = +25^\circ C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-2$ to $-V$</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>$R_O$</td>
<td>$f = 1kHz$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-28$ to $-m\Omega$</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>$I_{SC}$</td>
<td>$V_I = 35V$, $T_A = +25^\circ C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-230$ to $-mA$</td>
</tr>
<tr>
<td>Peak Current</td>
<td>$I_{PK}$</td>
<td>$T_J = +25^\circ C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$-2.2$ to $-A$</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Changes in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.
**Electrical Characteristics (MC7805A)**

(Refer to the test circuits. 0°C < TJ < 125°C, IO = 1A, VI = 10V, C = 0.33μF, C = 0.1μF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>VO</td>
<td>TI = +25°C</td>
<td>4.9</td>
<td>5</td>
<td>5.1</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 5mA to 1A, PPD ≤ 15W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 7.5V to 20V</td>
<td>4.8</td>
<td>5</td>
<td>5.2</td>
<td>V</td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Regline</td>
<td>VI = 7.5V to 25V</td>
<td>-</td>
<td>5</td>
<td>50</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 500mA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 8V to 12V</td>
<td>-</td>
<td>3</td>
<td>50</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TI = +25°C</td>
<td>-</td>
<td>5</td>
<td>50</td>
<td>mV</td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>TI = +25°C</td>
<td>-</td>
<td>9</td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 5mA to 1.5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 5mA to 1A</td>
<td>-</td>
<td>9</td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 5mA to 750mA</td>
<td>-</td>
<td>4</td>
<td>50</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>IQ</td>
<td>TJ = +25°C</td>
<td>-</td>
<td>5.0</td>
<td>6</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>ΔIQ</td>
<td>IO = 5mA to 1A</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 8V to 25V, IO = 500mA</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 7.5V to 20V, TJ = +25°C</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>mA</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>ΔV/ΔT</td>
<td>IO = 5mA</td>
<td>-</td>
<td>-0.8</td>
<td>-</td>
<td>mV/°C</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>VN</td>
<td>f = 10Hz to 100kHz, TA = +25°C</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>μV/Vo</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz, IO = 500mA</td>
<td>-</td>
<td>-68</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>VDrop</td>
<td>IO = 1A, TJ = +25°C</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>V</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>RO</td>
<td>f = 1kHz</td>
<td>-</td>
<td>17</td>
<td>-</td>
<td>mΩ</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>ISC</td>
<td>VI = 35V, TA = +25°C</td>
<td>-</td>
<td>250</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>IPk</td>
<td>TJ = +25°C</td>
<td>-</td>
<td>2.2</td>
<td>-</td>
<td>A</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Change in VO due to heating effects must be taken into account separately. Pulse testing with low duty is used.
### Electrical Characteristics (MC7806A)

(Refer to the test circuits. 0°C < TJ < 125°C, IO = 1A, V I = -11V, C I=0.33μF, C O=0.1μF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>VO</td>
<td>T J=+25°C</td>
<td>5.58</td>
<td>6</td>
<td>6.12</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 5mA to 1A, P0 ≤ 15W, VI = 8.6V to 21V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Regline</td>
<td>VI= 8.6V to 25V, IO = 500mA</td>
<td>-</td>
<td>5</td>
<td>60</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI= 9V to 13V, T J =+25°C</td>
<td>-</td>
<td>3</td>
<td>60</td>
<td>mV</td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>T J =+25°C, VI= 8.6V to 21V, IO = 6mA to 1.5A</td>
<td>-</td>
<td>5</td>
<td>60</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI= 9V to 13V, T J =+25°C</td>
<td>-</td>
<td>1.5</td>
<td>30</td>
<td>mV</td>
</tr>
<tr>
<td>Quescent Current</td>
<td>IQ</td>
<td>T J=+25°C</td>
<td>-</td>
<td>4.3</td>
<td>6</td>
<td>mA</td>
</tr>
<tr>
<td>Quescent Current Change</td>
<td>ΔIQ</td>
<td>IO = 5mA to 1A</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI= 9V to 25V, IO = 500mA</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI= 8.5V to 21V, T J =+25°C</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>mA</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>ΔV/1AT</td>
<td>IO = 5mA</td>
<td>-0.8</td>
<td>-</td>
<td></td>
<td>mV/°C</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>VN</td>
<td>T = 10kHz to 100kHz, TA =+25°C</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>μV/Vo</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz, IO = 500mA</td>
<td>-</td>
<td>65</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 9V to 19V</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>VDrop</td>
<td>IO = 1A, T J =+25°C</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>V</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>rO</td>
<td>T = 1kHz</td>
<td>-</td>
<td>17</td>
<td>-</td>
<td>mΩ</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>ISC</td>
<td>VI = 35V, TA =+25°C</td>
<td>-</td>
<td>250</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>IPK</td>
<td>T J=+25°C</td>
<td>-</td>
<td>2.2</td>
<td>-</td>
<td>A</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Change in VO due to heating effects must be taken into account separately. Pulse testing with low duty is used.
## Electrical Characteristics (MC7808A)

(Refer to the test circuits. 0°C < TJ < 125°C, IO = 1A, V1 = 14V, C = 0.33µF, C2 = 0.1µF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>VO</td>
<td>Tjis+25°C, IO = 5mA to 1A, PO ≤ 15W, V1 = 10.8V to 23V</td>
<td>7.7</td>
<td>8</td>
<td>8.16</td>
<td>V</td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Regline</td>
<td>V1 = 10.6V to 25V, IO = 500mA</td>
<td>7.7</td>
<td>8</td>
<td>8.3</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V1 = 11.1V to 17V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tjis+25°C, V1 = 10.4V to 23V</td>
<td>-</td>
<td>6</td>
<td>60</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V1 = 11V to 17V</td>
<td>-</td>
<td>3</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>Tjis+25°C, IO = 5mA to 1.5A</td>
<td>12</td>
<td></td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 5mA to 1A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IO = 250mA to 750mA</td>
<td>-</td>
<td>5</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>IQ</td>
<td>Tjis+25°C, IO = 6mA to 1A</td>
<td>5.0</td>
<td>6</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>ΔIQ</td>
<td>Tjis+25°C, IO = 6mA to 1A</td>
<td>-</td>
<td>-0.8</td>
<td>-</td>
<td>mV/°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V1 = 11V to 25V, IO = 500mA</td>
<td>-</td>
<td>-</td>
<td>0.6</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V1 = 10.6V to 23V, Tjis+25°C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>ΔV/ΔT</td>
<td>IO = 5mA</td>
<td>10</td>
<td>-0.8</td>
<td>-</td>
<td>mV/°C</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>VN</td>
<td>f = 10Hz to 100KHz, TA =+25°C</td>
<td>-</td>
<td>10</td>
<td></td>
<td>µV/VO</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz, IO = 600mA</td>
<td>-</td>
<td>62</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>Vdrop</td>
<td>f = 10Hz, IO = 1A, Tjis+25°C</td>
<td>-</td>
<td>2</td>
<td></td>
<td>VO</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>RO</td>
<td>f = 1KHz, TJ =+25°C</td>
<td>-</td>
<td>18</td>
<td></td>
<td>mΩ</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>ISC</td>
<td>V1 = 35V, TA =+25°C</td>
<td>-</td>
<td>250</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>IPK</td>
<td>TJ =+25°C</td>
<td>-</td>
<td>2.2</td>
<td></td>
<td>A</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Change in VO due to heating effects must be taken into account separately. Pulse testing with low duty is used.
### Electrical Characteristics (MC7809A)

(Refer to the test circuits. 0°C ≤ TJ ≤ 125°C, Iq = 1A, V = 15V, C = 0.33μF, C = 0.1μF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>VO</td>
<td>Tj = +25°C</td>
<td>8.82</td>
<td>9.0</td>
<td>9.16</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iq = 5mA to 1A, Pcm≤15W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 11.2V to 24V</td>
<td>8.65</td>
<td>9.0</td>
<td>9.35</td>
<td></td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Regline</td>
<td>Vi = 11.7V to 25V</td>
<td>-</td>
<td>6</td>
<td>90</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 12.5V to 19V</td>
<td>-</td>
<td>4</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 11.5V to 24V</td>
<td>-</td>
<td>6</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 12.5V to 19V</td>
<td>-</td>
<td>2</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>TJ = +25°C</td>
<td>12</td>
<td>100</td>
<td>-</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iq = 5mA to 1.0A</td>
<td>-</td>
<td>12</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iq = 5mA to 1.0A</td>
<td>-</td>
<td>12</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iq = 250mA to 750mA</td>
<td>-</td>
<td>5</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>IQ</td>
<td>TJ = +25°C</td>
<td>5.0</td>
<td>6.0</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>ΔIQ</td>
<td>Vi = 11.7V to 25V, TJ = +25°C</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 12V to 25V, Iq = 500mA</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iq = 5mA to 1.0A</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>AV/AT</td>
<td>Iq = 5mA</td>
<td>-</td>
<td>-1.0</td>
<td>-</td>
<td>mV/°C</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>VN</td>
<td>f = 10Hz to 100kHz, TA = +25°C</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>μV/Vo</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz, Iq = 500mA</td>
<td>-</td>
<td>62</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 12V to 22V</td>
<td>-</td>
<td>62</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Drop Out Voltage</td>
<td>VDrop</td>
<td>Iq = 1A, TJ = +25°C</td>
<td>-</td>
<td>2.0</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>f = 1kHz</td>
<td>-</td>
<td>17</td>
<td>-</td>
<td>mΩ</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>rO</td>
<td>f = 1kHz</td>
<td>-</td>
<td>17</td>
<td>-</td>
<td>mΩ</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>ISC</td>
<td>Vi = 35V, TA = +25°C</td>
<td>-</td>
<td>250</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>IPK</td>
<td>TJ = +25°C</td>
<td>-</td>
<td>2.2</td>
<td>-</td>
<td>A</td>
</tr>
</tbody>
</table>

Note:
1. Load and line regulation are specified at constant junction temperature. Change in VO due to heating effects must be taken into account separately. Pulse testing with low duty is used.

148
### Electrical Characteristics (MC7810A)
(Refer to the test circuits, $0^\circ C < T_J < 125^\circ C$, $I_O = 1A$, $V_I = 16V$, $C = 0.33\mu F$, $C_O = 0.1\mu F$, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>$V_O$</td>
<td>$T_J = \pm 25^\circ C$, $I_O = 5mA$ to $1A$, $P_O \leq 15W$</td>
<td>9.6</td>
<td>10</td>
<td>10.4</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 12.8V$ to $26V$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 500mA$</td>
<td>8</td>
<td>100</td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td>Line Regulation (Note 1)</td>
<td>Regline</td>
<td>$T_J = \pm 25^\circ C$, $I_O = 5mA$ to $1.5A$</td>
<td>-</td>
<td>8</td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 13V$ to $20V$</td>
<td>-</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 12.8V$ to $25V$</td>
<td>-</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 13V$ to $20V$</td>
<td>-</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load Regulation (Note 1)</td>
<td>Regload</td>
<td>$T_J = \pm 25^\circ C$, $I_O = 5mA$ to $1A$</td>
<td>-</td>
<td>12</td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 5mA$ to $1.5A$</td>
<td>-</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 5mA$ to $250mA$</td>
<td>-</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>$I_Q$</td>
<td>$T_J = \pm 25^\circ C$</td>
<td>-</td>
<td>5.0</td>
<td>6.0</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>$\Delta I_Q$</td>
<td>$V_I = 13V$ to $26V$, $T_J = \pm 25^\circ C$</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 12.8V$ to $25V$, $I_O = 500mA$</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 5mA$ to $1.0A$</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>$\Delta V_{\text{IN}}$</td>
<td>$I_O = 5mA$</td>
<td>-1.0</td>
<td></td>
<td>mV/°C</td>
<td></td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>$V_N$</td>
<td>$f = 10Hz$ to $100KHz$, $T_A = \pm 25^\circ C$</td>
<td>-</td>
<td>10</td>
<td></td>
<td>µV/°C</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>$RR$</td>
<td>$I_O = 500mA$, $f = 120Hz$, $V_I = 14V$ to $24V$</td>
<td>-</td>
<td>62</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>$V_{\text{Drop}}$</td>
<td>$I_O = 1A$, $T_J = \pm 25^\circ C$</td>
<td>-</td>
<td>2.0</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>$R_O$</td>
<td>$f = 1KHz$</td>
<td>-</td>
<td>17</td>
<td></td>
<td>µΩ</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>$I_{SC}$</td>
<td>$V_I = 35V$, $T_A = \pm 25^\circ C$</td>
<td>-</td>
<td>250</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>$I_{PK}$</td>
<td>$T_J = \pm 25^\circ C$</td>
<td>-</td>
<td>2.2</td>
<td></td>
<td>A</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Change in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.
### Electrical Characteristics (MC7812A)

(Refer to the test circuits. \(0 \degree C < T_J < 125 \degree C\), \(I_O = 1A\), \(V_i = 19V\), \(C = 0.33\mu F\), \(C_o = 0.1\mu F\), unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>(V_O)</td>
<td>(T_J = +25 \degree C)</td>
<td>11.75</td>
<td>12</td>
<td>12.25</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(I_O = 5mA) to 1A, (P_D \leq 15W)</td>
<td>11.5</td>
<td>12</td>
<td>12.5</td>
<td></td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>(V_{in})</td>
<td>(14.8V) to 30V</td>
<td>-</td>
<td>10</td>
<td>120</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(I_O = 500mA)</td>
<td>-</td>
<td>10</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(V_{in}) = 16V to 22V</td>
<td>-</td>
<td>4</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(T_J = +25 \degree C)</td>
<td>-</td>
<td>10</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(V_{in}) = 14.9V to 27V</td>
<td>-</td>
<td>3</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(V_{in}) = 16V to 22V</td>
<td>-</td>
<td>3</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>(V_{in})</td>
<td>(T_J = +25 \degree C)</td>
<td>-</td>
<td>12</td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(I_O = 5mA) to 1.5A</td>
<td>-</td>
<td>12</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(I_O = 5mA) to 1.0A</td>
<td>-</td>
<td>12</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(I_O = 250mA) to 750mA</td>
<td>-</td>
<td>5</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>(I_Q)</td>
<td>(T_J = +25 \degree C)</td>
<td>-</td>
<td>5.1</td>
<td>6.0</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>(\Delta I_Q)</td>
<td>(V_{in}) = 15V to 30V</td>
<td>-</td>
<td>0.8</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(T_J = +25 \degree C)</td>
<td>-</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(V_{in}) = 14V to 27V, (I_O = 800mA)</td>
<td>-</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(T_J = +25 \degree C)</td>
<td>-</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>(\Delta V_{in}/T)</td>
<td>(I_O = 5mA)</td>
<td>-</td>
<td>1.0</td>
<td></td>
<td>mV/°C</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>(V_N)</td>
<td>(f = 10Hz) to 100KHz, (T_A = +25 \degree C)</td>
<td>-</td>
<td>10</td>
<td></td>
<td>μV/Vo</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>(R_R)</td>
<td>(f = 120Hz, I_O = 500mA)</td>
<td>-</td>
<td>60</td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(V_{in}) = 14V to 24V</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>(V_{Drop})</td>
<td>(I_O = 1A, T_J = +25 \degree C)</td>
<td>-</td>
<td>2.0</td>
<td>-</td>
<td>V</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>(r_O)</td>
<td>(f = 1kHz)</td>
<td>-</td>
<td>18</td>
<td>-</td>
<td>mΩ</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>(I_{SC})</td>
<td>(V_{in}) = 35V, (T_A = +25 \degree C)</td>
<td>-</td>
<td>250</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>(I_{PK})</td>
<td>(T_J = +25 \degree C)</td>
<td>-</td>
<td>2.2</td>
<td>-</td>
<td>A</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Change in \(V_O\) due to heating effects must be taken into account separately. Pulse testing with low duty cycle is used.
## Electrical Characteristics (MC7815A)

(Refer to the test circuits. 0°C < Tj < 125°C, Io = 1A, Vi = 23V, C = 0.33μF, C = 0.1μF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>V0</td>
<td>Tj = +25°C</td>
<td>14.7</td>
<td>15</td>
<td>15.3</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Io = 5mA to 1A, Pd ≤ 16W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 17.7V to 30V</td>
<td>14.4</td>
<td>15</td>
<td>15.6</td>
<td></td>
</tr>
<tr>
<td>Line Regulation (Note 1)</td>
<td>Regline</td>
<td>Vi = 17.9V to 30V</td>
<td>-</td>
<td>10</td>
<td>150</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Io = 500mA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 20V to 26V</td>
<td>-</td>
<td>5</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tj = +25°C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 17.5V to 30V</td>
<td>-</td>
<td>11</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 20V to 26V</td>
<td>-</td>
<td>3</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>Load Regulation (Note 1)</td>
<td>Regload</td>
<td>Tj = +25°C</td>
<td>-</td>
<td>12</td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Io = 5mA to 1.5A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Io = 5mA to 1.0A</td>
<td>-</td>
<td>12</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Io = 5mA to 0.1A</td>
<td>-</td>
<td>5</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Io = 250mA to 760mA</td>
<td>-</td>
<td>5.2</td>
<td>6.0</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>Io</td>
<td>Tj = +25°C</td>
<td>-</td>
<td>5.2</td>
<td>6.0</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>△Io</td>
<td>Vi = 17.5V to 30V, Tj = +25°C</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 17.5V to 30V, Io = 500mA</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Io = 5mA to 1.0A</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>△V/△IT</td>
<td>Io = 5mA</td>
<td>-1.0</td>
<td>-</td>
<td>-</td>
<td>mV/°C</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>VN</td>
<td>f = 10Hz to 100KHz</td>
<td>-10</td>
<td>-</td>
<td>-</td>
<td>μV/V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TA = +25°C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz, Io = 500mA</td>
<td>58</td>
<td>-</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 18.5V to 28.5V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>VDrop</td>
<td>Io = 1A, Tj = +25°C</td>
<td>2.0</td>
<td>-</td>
<td>-</td>
<td>V</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>rO</td>
<td>f = 1kHz</td>
<td>19</td>
<td>-</td>
<td>-</td>
<td>mΩ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vi = 35V, TA = +25°C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>ISC</td>
<td>Vi = 35V, Tj = +25°C</td>
<td>250</td>
<td>-</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>IPK</td>
<td>Tj = +25°C</td>
<td>2.2</td>
<td>-</td>
<td>-</td>
<td>A</td>
</tr>
</tbody>
</table>

**Note:**
1. Load and line regulation are specified at constant junction temperature. Change in V0 due to heating effects must be taken into account separately. Pulse testing with low duty is used.
## Electrical Characteristics (MC7818A)

(Refer to the test circuits. $0 \text{°C} < T_J < 125 \text{°C}$, $I_O = 1A$, $V_I = 27V$, $C = 0.33 \mu F$, $C_O = 0.1 \mu F$, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>$V_O$</td>
<td>$T_J = +25 \text{°C}$</td>
<td>17.64</td>
<td>18</td>
<td>18.36</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 5 \text{mA to 1A}$, $P_O \leq 15W$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 21V$ to $33V$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Regline</td>
<td>$V_I = 21V$ to $33V$</td>
<td>-</td>
<td>15</td>
<td>180</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 500 \text{mA}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_J = +25 \text{°C}$</td>
<td>-</td>
<td>5</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 21V$ to $33V$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>$T_J = +25 \text{°C}$</td>
<td>15</td>
<td>15</td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 5 \text{mA to 1.5A}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 5 \text{mA to 1.0A}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 250 \text{mA to 750mA}$</td>
<td>-</td>
<td>7</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>$I_Q$</td>
<td>$T_J = +25 \text{°C}$</td>
<td>-</td>
<td>5.2</td>
<td>6.0</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>$\Delta I_Q$</td>
<td>$V_I = 21V$ to $33V$, $T_J = +25 \text{°C}$</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V_I = 21V$ to $33V$, $I_O = 500 \text{mA}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I_O = 5 \text{mA to 1.0A}$</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td>mA</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>$\Delta V/IAT$</td>
<td>$I_O = 5 \text{mA}$</td>
<td>-</td>
<td>-1.0</td>
<td>-</td>
<td>mV/°C</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>$V_N$</td>
<td>$f = 10 \text{Hz}$ to $100\text{KHz}$</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>μV/°C</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>$R_R$</td>
<td>$f = 120\text{Hz}$, $I_O = 500 \text{mA}$</td>
<td>-</td>
<td>57</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>$V_{Drop}$</td>
<td>$I_O = 1A$, $T_J = +25 \text{°C}$</td>
<td>-</td>
<td>2.0</td>
<td>-</td>
<td>V</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>$r_O$</td>
<td>$f = 1\text{KHz}$</td>
<td>-</td>
<td>19</td>
<td>-</td>
<td>mΩ</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>$I_{SC}$</td>
<td>$V_I = 35V$, $T_J = +25 \text{°C}$</td>
<td>-</td>
<td>250</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>$I_{PK}$</td>
<td>$T_J = +25 \text{°C}$</td>
<td>-</td>
<td>2.2</td>
<td>-</td>
<td>A</td>
</tr>
</tbody>
</table>

**Note:**

1. Load and line regulation are specified at constant junction temperature. Change in $V_O$ due to heating effects must be taken into account separately. Pulse testing with low duty is used.
### Electrical Characteristics (MC7824A)

(Refer to the test circuits. 0°C < TJ < 125°C, IO = 1A, VI = 33V, C = 0.33µF, C = 0.1µF, unless otherwise specified)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Conditions</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>VO</td>
<td>TJ = +25°C</td>
<td>23.5</td>
<td>24</td>
<td>24.5</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IQ = 5mA to 1A, PO ≤ 15W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 27V to 38V</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 21V to 33V</td>
<td>-</td>
<td>6</td>
<td>240</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 26.7V to 38V</td>
<td>-</td>
<td>6</td>
<td>240</td>
<td>mV</td>
</tr>
<tr>
<td>Line Regulation (Note1)</td>
<td>Regline</td>
<td>VI = 27V to 38V</td>
<td>16</td>
<td></td>
<td>240</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 21V to 33V</td>
<td>-</td>
<td>6</td>
<td>240</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 26.7V to 38V</td>
<td>-</td>
<td>6</td>
<td>240</td>
<td>mV</td>
</tr>
<tr>
<td>Load Regulation (Note1)</td>
<td>Regload</td>
<td>TJ = +25°C</td>
<td>15</td>
<td></td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IQ = 5mA to 1.5A</td>
<td>-</td>
<td>15</td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IQ = 5mA to 1.0A</td>
<td>-</td>
<td>15</td>
<td>100</td>
<td>mV</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IQ = 250mA to 750mA</td>
<td>5</td>
<td>7</td>
<td>50</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current</td>
<td>IQ</td>
<td>TJ = +25°C</td>
<td>5.2</td>
<td></td>
<td>6.0</td>
<td>mA</td>
</tr>
<tr>
<td>Quiescent Current Change</td>
<td>ΔIQ</td>
<td>VI = 27.3V to 38V, TJ = +25°C</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VI = 27.3V to 38V, IQ = 500mA</td>
<td>-</td>
<td>-</td>
<td>0.8</td>
<td>mA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IQ = 5mA to 1.0A</td>
<td>-</td>
<td>-</td>
<td>0.5</td>
<td>mA</td>
</tr>
<tr>
<td>Output Voltage Drift</td>
<td>ΔV/ΔT</td>
<td>IQ = 5mA</td>
<td>-</td>
<td>-1.5</td>
<td>-</td>
<td>mV/C</td>
</tr>
<tr>
<td>Output Noise Voltage</td>
<td>VN</td>
<td>f = 10Hz to 100KHz</td>
<td>-</td>
<td>10</td>
<td>-</td>
<td>µV/µA</td>
</tr>
<tr>
<td>Ripple Rejection</td>
<td>RR</td>
<td>f = 120Hz, IQ = 500mA</td>
<td>-</td>
<td>54</td>
<td>-</td>
<td>dB</td>
</tr>
<tr>
<td>Dropout Voltage</td>
<td>VDrop</td>
<td>IQ = 1A, TJ = +25°C</td>
<td>-</td>
<td>2.0</td>
<td>-</td>
<td>V</td>
</tr>
<tr>
<td>Output Resistance</td>
<td>RO</td>
<td>f = 1KHz</td>
<td>-</td>
<td>20</td>
<td>-</td>
<td>mΩ</td>
</tr>
<tr>
<td>Short Circuit Current</td>
<td>ISC</td>
<td>VI = 35V, TA = +25°C</td>
<td>-</td>
<td>250</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>Peak Current</td>
<td>IPK</td>
<td>TJ = +25°C</td>
<td>-</td>
<td>2.2</td>
<td>-</td>
<td>A</td>
</tr>
</tbody>
</table>

**Note:**

1. Load and line regulation are specified at constant junction temperature. Change in VO due to heating effects must be taken into account separately. Pulse testing with low duty is used.
Typical Performance Characteristics

Figure 1. Quiescent Current

Figure 2. Peak Output Current

Figure 3. Output Voltage

Figure 4. Quiescent Current
Typical Applications

Figure 6. DC Parameters

Figure 6. Load Regulation

Figure 7. Ripple Rejection

Figure 8. Fixed Output Regulator
Figure 9. Constant Current Regulator

Notes:
1. To specify an output voltage, substitute voltage value for "XX." A common ground is required between the input and the output voltage. The input voltage must remain typically 2.0V above the output voltage even during the low point on the input ripple voltage.
2. C1 is required if regulator is located an appreciable distance from power supply filter.
3. C0 improves stability and transient response.

Figure 10. Circuit for Increasing Output Voltage

Figure 11. Adjustable Output Regulator (7 to 30V)
Figure 12. High Current Voltage Regulator

Figure 13. High Output Current with Short Circuit Protection

Figure 14. Tracking Voltage Regulator
Figure 15. Split Power Supply (±15V-1A)

Figure 16. Negative Output Voltage Circuit

Figure 17. Switching Regulator
Mechanical Dimensions

Package

TO-220
Mechanical Dimensions (Continued)

Package

D-PAK

[Diagrams showing dimensions of the D-PAK package]
## Ordering Information

<table>
<thead>
<tr>
<th>Product Number</th>
<th>Output Voltage Tolerance</th>
<th>Package</th>
<th>Operating Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>LM7805CT</td>
<td>±4%</td>
<td>TO-220</td>
<td>0 ~ +125°C</td>
</tr>
<tr>
<td>MC7805CT</td>
<td>±4%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7806CT</td>
<td>±4%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7808CT</td>
<td>±4%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7809CT</td>
<td>±4%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7810CT</td>
<td>±4%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7812CT</td>
<td>±4%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7815CT</td>
<td>±4%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7818CT</td>
<td>±4%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7824CT</td>
<td>±4%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7805CDT</td>
<td>±4%</td>
<td>D-PAK</td>
<td>0 ~ +125°C</td>
</tr>
<tr>
<td>MC7806CDT</td>
<td>±4%</td>
<td>D-PAK</td>
<td></td>
</tr>
<tr>
<td>MC7808CDT</td>
<td>±4%</td>
<td>D-PAK</td>
<td></td>
</tr>
<tr>
<td>MC7809CDT</td>
<td>±4%</td>
<td>D-PAK</td>
<td></td>
</tr>
<tr>
<td>MC7810CDT</td>
<td>±4%</td>
<td>D-PAK</td>
<td></td>
</tr>
<tr>
<td>MC7812CDT</td>
<td>±4%</td>
<td>D-PAK</td>
<td></td>
</tr>
<tr>
<td>MC7805ACT</td>
<td>±2%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7806ACT</td>
<td>±2%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7808ACT</td>
<td>±2%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7809ACT</td>
<td>±2%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7810ACT</td>
<td>±2%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7812ACT</td>
<td>±2%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7815ACT</td>
<td>±2%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7818ACT</td>
<td>±2%</td>
<td>TO-220</td>
<td></td>
</tr>
<tr>
<td>MC7824ACT</td>
<td>±2%</td>
<td>TO-220</td>
<td></td>
</tr>
</tbody>
</table>
DISCLAIMER
FAIRCHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HERIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

LIFE SUPPORT POLICY
FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:
1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.
2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

www.fairchildsemi.com

© 2001 Fairchild Semiconductor Corporation
This datasheet has been download from:

www.datasheetcatalog.com

Datasheets for electronics components.
5.5 Arduino Mega1280

**Features**
- **High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller**
- **Advanced RISC Architecture**
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- **High Endurance Non-volatile Memory Segments**
  - 8 Kbytes EEPROM
  - 512 Kbytes SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
- **In-System Programming by On-chip Boot Program**
- **True Read-While-Write Operation**
- **Programming Lock for Software Security**
- **Endurance: Up to 64 Kbytes Optional External Memory Space**
- **JTAG (IEEE std. 1149.1 compliant) Interface**
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- **Peripheral Features**
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial UART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- **Special Microcontroller Features**
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- **I/O and Packages**
  - 54/88 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pin QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- **Temperature Range:**
  - −40°C to 85°C Industrial
- **Ultra-Low Power Consumption**
  - Active Mode: 1 MHz, 1.8V: 500 µA
  - Power-down Mode: 0.1 µA at 1.8V
- **Speed Grade:**
  - ATmega640/ATmega1280/ATmega2560:
    - 0 to 4 MHz @ 1.8V - 5.5V, 0 to 8 MHz @ 2.7V - 5.5V
  - ATmega2560/ATmega512V:
    - 0 to 2 MHz @ 1.8V - 5.5V, 0 to 8 MHz @ 2.7V - 5.5V
  - ATmega640/ATmega1280/ATmega2561:
    - 0 to 8 MHz @ 2.7V - 5.5V, 0 to 16 MHz @ 4.5V - 5.5V
  - ATmega2560/ATmega512I:
    - 0 to 16 MHz @ 4.5V - 5.5V
1. Pin Configurations

Figure 1-1. TQFP-pinout ATmega640/1280/2560/2561
Table 1-1. CBGA-pinout ATmega640/1280/2560

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>GND</td>
<td>AREF</td>
<td>PF0</td>
<td>PF2</td>
<td>PF5</td>
<td>PK0</td>
<td>PK3</td>
<td>PK6</td>
<td>GND</td>
<td>VCC</td>
</tr>
<tr>
<td>B</td>
<td>AVCC</td>
<td>PG5</td>
<td>PF1</td>
<td>PF3</td>
<td>PF6</td>
<td>PK1</td>
<td>PK4</td>
<td>PK7</td>
<td>PA0</td>
<td>PA2</td>
</tr>
<tr>
<td>C</td>
<td>PE2</td>
<td>PE0</td>
<td>PE1</td>
<td>PF4</td>
<td>PF7</td>
<td>PK2</td>
<td>PK5</td>
<td>PJ7</td>
<td>PA1</td>
<td>PA3</td>
</tr>
<tr>
<td>D</td>
<td>PE3</td>
<td>PE4</td>
<td>PE5</td>
<td>PE6</td>
<td>PH2</td>
<td>PA4</td>
<td>PA5</td>
<td>PA6</td>
<td>PA7</td>
<td>PG2</td>
</tr>
<tr>
<td>E</td>
<td>PE7</td>
<td>PH0</td>
<td>PH1</td>
<td>PH3</td>
<td>PH5</td>
<td>PJ6</td>
<td>PJ5</td>
<td>PJ4</td>
<td>PJ3</td>
<td>PJ2</td>
</tr>
<tr>
<td>F</td>
<td>VCC</td>
<td>PH4</td>
<td>PH6</td>
<td>PB0</td>
<td>PL4</td>
<td>PD1</td>
<td>PJ1</td>
<td>PJ0</td>
<td>PC7</td>
<td>GND</td>
</tr>
<tr>
<td>G</td>
<td>GND</td>
<td>PB1</td>
<td>PB2</td>
<td>PB5</td>
<td>PL2</td>
<td>PD0</td>
<td>PD5</td>
<td>PC5</td>
<td>PC6</td>
<td>VCC</td>
</tr>
<tr>
<td>H</td>
<td>PB3</td>
<td>PB4</td>
<td>RESET</td>
<td>PL1</td>
<td>PL3</td>
<td>PL7</td>
<td>PD4</td>
<td>PC4</td>
<td>PC3</td>
<td>PC2</td>
</tr>
<tr>
<td>J</td>
<td>PH7</td>
<td>PG3</td>
<td>PB6</td>
<td>PL0</td>
<td>XTAL2</td>
<td>PL6</td>
<td>PD3</td>
<td>PC1</td>
<td>PC0</td>
<td>PG1</td>
</tr>
<tr>
<td>K</td>
<td>PB7</td>
<td>PG4</td>
<td>VCC</td>
<td>GND</td>
<td>XTAL1</td>
<td>PL5</td>
<td>PD2</td>
<td>PD6</td>
<td>PD7</td>
<td>PG0</td>
</tr>
</tbody>
</table>

Note: The functions for each pin is the same as for the 100 pin packages shown in Figure 1-1 on page 2.
Figure 1-3. Pinout ATmega1281/2561

Note: The large center pad underneath the QFN/MLF package is made of metal and internally connected to GND. It should be soldered or glued to the board to ensure good mechanical stability. If the center pad is left unconnected, the package might loosen from the board.
2. Overview

The ATmega640/1280/1281/2560/2561 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega640/1280/1281/2560/2561 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram

NOTE:
Shaded parts only available in the 100-pin version.

Complete functionality for the ADC, T/C4, and T/C5 only available in the 100-pin version.
The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega640/1280/1281/2560/2561 provides the following features: 64K/128K/256K bytes of In-System Programmable Flash with Read-While-Write capabilities, 4 Kbytes EEPROM, 8 Kbytes SRAM, 54/86 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), six flexible Timer/Counters with compare modes and PWM, 4 USARTs, a byte oriented 2-wire Serial Interface, a 16-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with Internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the Crystal/Resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel’s high-density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega640/1280/1281/2560/2561 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega640/1280/1281/2560/2561 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.
2.2 Comparison Between ATmega1281/2561 and ATmega640/1280/2560

Each device in the ATmega640/1280/1281/2560/2561 family differs only in memory size and number of pins. Table 2-1 summarizes the different configurations for the six devices.

### Table 2-1. Configuration Summary

<table>
<thead>
<tr>
<th>Device</th>
<th>Flash</th>
<th>EEPROM</th>
<th>RAM</th>
<th>General Purpose I/O pins</th>
<th>16 bits resolution PWM channels</th>
<th>Serial USARTs</th>
<th>ADC Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATmega640</td>
<td>64KB</td>
<td>4KB</td>
<td>8KB</td>
<td>86</td>
<td>12</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>ATmega1280</td>
<td>128KB</td>
<td>4KB</td>
<td>8KB</td>
<td>86</td>
<td>12</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>ATmega1281</td>
<td>128KB</td>
<td>4KB</td>
<td>8KB</td>
<td>54</td>
<td>6</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>ATmega2560</td>
<td>256KB</td>
<td>4KB</td>
<td>8KB</td>
<td>86</td>
<td>12</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>ATmega2561</td>
<td>256KB</td>
<td>4KB</td>
<td>8KB</td>
<td>54</td>
<td>6</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

2.3 Pin Descriptions

2.3.1 VCC

Digital supply voltage.

2.3.2 GND

Ground.

2.3.3 Port A (PA7..PA0)

Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 78.

2.3.4 Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B has better driving capabilities than the other ports.

Port B also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 79.

2.3.5 Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up
resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of special features of the ATmega640/1280/1281/2560/2561 as listed on page 82.

2.3.6 Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 83.

2.3.7 Port E (PE7..PE0)

Port E is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 86.

2.3.8 Port F (PF7..PF0)

Port F serves as analog inputs to the A/D Converter.

Port F also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a reset occurs.

Port F also serves the functions of the JTAG interface.

2.3.9 Port G (PG5..PG0)

Port G is a 6-bit I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 90.

2.3.10 Port H (PH7..PH0)

Port H is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port H output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port H pins that are externally pulled low will source current if the pull-up
resistors are activated. The Port H pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port H also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 92.

2.3.11 Port J (PJ7..PJ0)

Port J is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port J output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port J pins that are externally pulled low will source current if the pull-up resistors are activated. The Port J pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port J also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 94.

2.3.12 Port K (PK7..PK0)

Port K serves as analog inputs to the A/D Converter.

Port K is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port K output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port K pins that are externally pulled low will source current if the pull-up resistors are activated. The Port K pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port K also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 96.

2.3.13 Port L (PL7..PL0)

Port L is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port L output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port L pins that are externally pulled low will source current if the pull-up resistors are activated. The Port L pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port L also serves the functions of various special features of the ATmega640/1280/2560 as listed on page 98.

2.3.14 RESET

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in “System and Reset Characteristics” on page 372. Shorter pulses are not guaranteed to generate a reset.

2.3.15 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

2.3.16 XTAL2

Output from the inverting Oscillator amplifier.
2.3.17 AVCC

AVCC is the supply voltage pin for Port F and the A/D Converter. It should be externally connected to $V_{CC}$, even if the ADC is not used. If the ADC is used, it should be connected to $V_{CC}$ through a low-pass filter.

2.3.18 AREF

This is the analog reference pin for the A/D Converter.
3. Resources

A comprehensive set of development tools and application notes, and datasheets are available for download on http://www.atmel.com/avr.

4. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Please confirm with the C compiler documentation for more details.

These code examples assume that the part specific header file is included before compilation. For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

5. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 ppm over 20 years at 85°C or 100 years at 25°C.
6. AVR CPU Core

6.1 Introduction
This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

6.2 Architectural Overview

Figure 6-1. Block Diagram of the AVR Architecture

In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.
The fast-access Register File contains 32 × 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16-bit or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, the ATmega640/1280/1281/2560/2561 has Extended I/O space from 0x60 - 0x1FF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

### 6.3 ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.
6.4 Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

6.4.1 SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3F (0x5F)</td>
<td>I</td>
<td>T</td>
<td>H</td>
<td>S</td>
<td>V</td>
<td>N</td>
<td>Z</td>
<td>C</td>
</tr>
</tbody>
</table>

**Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

**Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit Load) and BST (Bit Store) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

**Bit 5 – H: Half Carry Flag**

The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry is useful in BCD arithmetic. See the “Instruction Set Description” for detailed information.

**Bit 4 – S: Sign Bit, S = N ⊕ V**

The S-bit is always an exclusive or between the Negative Flag N and the Two’s Complement Overflow Flag V. See the “Instruction Set Description” for detailed information.

**Bit 3 – V: Two’s Complement Overflow Flag**

The Two’s Complement Overflow Flag V supports two’s complement arithmetics. See the “Instruction Set Description” for detailed information.

**Bit 2 – N: Negative Flag**

The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

**Bit 1 – Z: Zero Flag**

The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.
• Bit 0 – C: Carry Flag
The Carry Flag C indicates a carry in an arithmetic or logic operation. See the “Instruction Set Description” for detailed information.

6.5 General Purpose Register File
The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:
• One 8-bit output operand and one 8-bit result input
• Two 8-bit output operands and one 8-bit result input
• Two 8-bit output operands and one 16-bit result input
• One 16-bit output operand and one 16-bit result input
Figure 6-2 shows the structure of the 32 general purpose working registers in the CPU.

Figure 6-2. AVR CPU General Purpose Working Registers

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>Addr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>0x02</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R13</td>
<td>0x0D</td>
<td></td>
</tr>
<tr>
<td>General</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R14</td>
<td>0x0E</td>
<td></td>
</tr>
<tr>
<td>R15</td>
<td>0x0F</td>
<td></td>
</tr>
<tr>
<td>Purpose</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R16</td>
<td>0x10</td>
<td></td>
</tr>
<tr>
<td>Working</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R17</td>
<td>0x11</td>
<td></td>
</tr>
<tr>
<td>Registers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R26</td>
<td>0x1A</td>
<td>X-register Low Byte</td>
</tr>
<tr>
<td>R27</td>
<td>0x1B</td>
<td>X-register High Byte</td>
</tr>
<tr>
<td>R28</td>
<td>0x1C</td>
<td>Y-register Low Byte</td>
</tr>
<tr>
<td>R29</td>
<td>0x1D</td>
<td>Y-register High Byte</td>
</tr>
<tr>
<td>R30</td>
<td>0x1E</td>
<td>Z-register Low Byte</td>
</tr>
<tr>
<td>R31</td>
<td>0x1F</td>
<td>Z-register High Byte</td>
</tr>
</tbody>
</table>

Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 6-2, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y- and Z-pointer registers can be set to index any register in the file.

6.5.1 The X-register, Y-register, and Z-register
The registers R26..R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in Figure 6-3 on page 16.
Figure 6-3. The X-, Y-, and Z-registers

In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the instruction set reference for details).

6.6 Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above 0x0200. The initial value of the stack pointer is the last address of the internal SRAM. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two for ATmega640/1280/1281 and three for ATmega2560/2561 when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two for ATmega640/1280/1281 and three for ATmega2560/2561 when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.
6.6.1 RAMPZ – Extended Z-pointer Register for ELPM/SPM

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3B (0x5B)</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

Read/Write

Initial Value

For ELPM/SPM instructions, the Z-pointer is a concatenation of RAMPZ, ZH, and ZL, as shown in Figure 6-4. Note that LPM is not affected by the RAMPZ setting.

Figure 6-4. The Z-pointer used by ELPM and SPM

<table>
<thead>
<tr>
<th>Bit (Individually)</th>
<th>7</th>
<th>0</th>
<th>7</th>
<th>0</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
</table>

RAMPZ ZH ZL

Bit (Z-pointer)

23 16 15 8 7 0

The actual number of bits is implementation dependent. Unused bits in an implementation will always read as zero. For compatibility with future devices, be sure to write these bits to zero.

6.6.2 EIND – Extended Indirect Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3C (0x5C)</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
</tbody>
</table>

Read/Write

Initial Value

For EICALL/EIJMP instructions, the Indirect-pointer to the subroutine/routine is a concatenation of EIND, ZH, and ZL, as shown in Figure 6-5. Note that ICALL and IJMP are not affected by the EIND setting.

Figure 6-5. The Indirect-pointer used by EICALL and EIJMP

<table>
<thead>
<tr>
<th>Bit (Individually)</th>
<th>7</th>
<th>0</th>
<th>7</th>
<th>0</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
</table>

EIND ZH ZL

Bit (Indirect-pointer)

23 16 15 8 7 0

The actual number of bits is implementation dependent. Unused bits in an implementation will always read as zero. For compatibility with future devices, be sure to write these bits to zero.

6.7 Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_cpu, directly generated from the selected clock source for the chip. No internal clock division is used.

Figure 6-6 on page 18 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.
6.8 Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security. See the section “Memory Programming” on page 335 for details.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. The complete list of vectors is shown in “Interrupts” on page 105. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INTO – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the MCU Control Register (MCUCR). Refer to “Interrupts” on page 105 for more information. The Reset Vector can also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse, see “Memory Programming” on page 335.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.
There are basically two types of interrupts. The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.

When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

### Assembly Code Example

```assembly
in r16, SREG ; store SREG value
cli ; disable interrupts during timed sequence
sbi EECR, EEMPE ; start EEPROM write
sbi EECR, EEPE
out SREG, r16 ; restore SREG value (I-bit)
```

### C Code Example

```c
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
__disable_interrupt();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR &= ~(1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.
Assembly Code Example

```assembly
sei ; set Global Interrupt Enable
sleep; enter sleep, waiting for interrupt
; note: will enter sleep before any pending
; interrupt(s)
```

C Code Example

```c
enable_interrupt(); /* set Global Interrupt Enable */
__sleep(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

6.8.1 Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is five clock cycles minimum. After five clock cycles the program vector address for the actual interrupt handling routine is executed. During these five clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by five clock cycles. This increase comes in addition to the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes five clock cycles. During these five clock cycles, the Program Counter (three bytes) is popped back from the Stack, the Stack Pointer is incremented by three, and the I-bit in SREG is set.
7. AVR Memories

This section describes the different memories in the ATmega640/1280/1281/2560/2561. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the ATmega640/1280/1281/2560/2561 features an EEPROM Memory for data storage. All three memory spaces are linear and regular.

7.1 In-System Reprogrammable Flash Program Memory

The ATmega640/1280/1281/2560/2561 contains 64K/128K/256K bytes On-chip In-System Reprogrammable Flash memory for program storage, see Figure 7-1. Since all AVR instructions are 16 bit or 32 bit wide, the Flash is organized as 32K/64K/128K × 16. For software security, the Flash Program memory space is divided into two sections, Boot Program section and Application Program section.

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega640/1280/1281/2560/2561 Program Counter (PC) is 15/16/17 bits wide, thus addressing the 32K/64K/128K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in "Boot Loader Support – Read-While-Write Self-Programming" on page 317. "Memory Programming" on page 335 contains a detailed description on Flash data serial downloading using the SPI pins or the JTAG interface.

Constant tables can be allocated within the entire program memory address space (see the LPM – Load Program Memory instruction description and ELPM - Extended Load Program Memory instruction description).

Timing diagrams for instruction fetch and execution are presented in "Instruction Execution Timing" on page 17.

Figure 7-1. Program Flash Memory Map

<table>
<thead>
<tr>
<th>Address (HEX)</th>
<th>0x0000/0x0000/0x0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Flash Section</td>
<td>0x0000/0x0000/0x0000</td>
</tr>
<tr>
<td>Boot Flash Section</td>
<td>0x7FFF/0xFFFF/0x1FFFF</td>
</tr>
</tbody>
</table>

7.2 SRAM Data Memory

Figure 7-2 on page 23 shows how the ATmega640/1280/1281/2560/2561 SRAM Memory is organized.

The ATmega640/1280/1281/2560/2561 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from $060 - $1FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The first 4,608/8,704 Data Memory locations address both the Register File, the I/O Memory, Extended I/O Memory, and the internal data SRAM. The first 32 locations address the Register file, the next 64 location the standard I/O Memory, then 416 locations of Extended I/O memory and the next 8,192 locations address the internal data SRAM.
An optional external data SRAM can be used with the ATmega640/1280/1281/2560/2561. This SRAM will occupy an area in the remaining address locations in the 64K address space. This area starts at the address following the internal SRAM. The Register file, I/O, Extended I/O and Internal SRAM occupies the lowest 4,608/8,704 bytes, so when using 64 Kbytes (65,536 bytes) of External Memory, 60,478/56,832 Bytes of External Memory are available. See “External Memory Interface” on page 28 for details on how to take advantage of the external memory map.

When the addresses accessing the SRAM memory space exceeds the internal data memory locations, the external data SRAM is accessed using the same instructions as for the internal data memory access. When the internal data memories are accessed, the read and write strobe pins (PG0 and PG1) are inactive during the whole access cycle. External SRAM operation is enabled by setting the SRE bit in the XMCRA Register.

Accessing external SRAM takes one additional clock cycle per byte compared to access of the internal SRAM. This means that the commands LD, ST, LDS, STS, LDD, STD, PUSH, and POP take one additional clock cycle. If the Stack is placed in external SRAM, interrupts, subroutine calls and returns take three clock cycles extra because the three-byte program counter is pushed and popped, and external memory access does not take advantage of the internal pipeline memory access. When external SRAM interface is used with wait-state, one-byte external access takes two, three, or four additional clock cycles for one, two, and three wait-states respectively. Interrupts, subroutine calls and returns will need five, seven, or nine clock cycles more than specified in the instruction set manual for one, two, and three wait-states.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register file, registers R26 to R31 feature the indirect addressing pointer registers.

The direct addressing reaches the entire data space.

The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.

The 32 general purpose working registers, 64 I/O registers, and the 4,196/8,192 bytes of internal data SRAM in the ATmega640/1280/1281/2560/2561 are all accessible through all these addressing modes. The Register File is described in “General Purpose Register File” on page 15.
7.2.1 Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clk\textsubscript{CPU} cycles as described in Figure 7-3.

Figure 7-3. On-chip Data SRAM Access Cycles

7.3 EEPROM Data Memory

The ATmega640/1280/1281/2560/2561 contains 4 Kbytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

For a detailed description of SPI, JTAG and Parallel data downloading to the EEPROM, see “Serial Downloading” on page 349, “Programming via the JTAG Interface” on page 354, and “Programming the EEPROM” on page 343 respectively.
7.3.1 EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space, see “Register Description” on page 35.

The write access time for the EEPROM is given in Table 7-1. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, \( V_{CC} \) is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See “Preventing EEPROM Corruption” on page 26, for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. See the description of the EEPROM Control Register for details on this, “Register Description” on page 35.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

The calibrated Oscillator is used to time the EEPROM accesses. Table 7-1 lists the typical programming time for EEPROM access from the CPU.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Number of Calibrated RC Oscillator Cycles</th>
<th>Typ Programming Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEPROM write (from CPU)</td>
<td>26,368</td>
<td>3.3 ms</td>
</tr>
</tbody>
</table>

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (for example by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.
### Assembly Code Example

```assembly
    EEPROM_write:
        ; Wait for completion of previous write
        sbic EECR,EEPE
        rjmp EEPROM_write
        ; Set up address (r18:r17) in address register
        out EEARH, r18
        out EEARL, r17
        ; Write data (r16) to Data Register
        out EEDR,r16
        ; Write logical one to EEMPE
        sbi EECR,EEMPE
        ; Start eeprom write by setting EEPE
        sbi EECR,EEPE
        ret
```

### C Code Example

```c
    void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
    {
        /* Wait for completion of previous write */
        while(EECR & (1<<EEPE))
        {
            ; /* Set up address and Data Registers */
            EEAR = uiAddress;
            EEDR = ucData;
            /* Write logical one to EEMPE */
            EECR |= (1<<EEMPE);
            /* Start eeprom write by setting EEPE */
            EECR |= (1<<EEPE);
        }
    }
```

Note: 1. See “About Code Examples” on page 11.
The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

### Assembly Code Example

```assembly
Assembly Code Example[1]

EEPROM_read:
    ; Wait for completion of previous write
    sbic EECR,EEPE
    rjmp EEPROM_read

    ; Set up address (r18:r17) in address register
    out EEARH, r18
    out EEARL, r17

    ; Start eeprom read by writing EERE
    sbi EECR,EERE

    ; Read data from Data Register
    in r16,EEDR

    ret
```

### C Code Example

```c
C Code Example[1]

unsigned char EEPROM_read(unsigned int uiAddress)
{
    /* Wait for completion of previous write */
    while(EECR & (1<EEPE));

    /* Set up address register */
    EEAR = uiAddress;

    /* Start eeprom read by writing EERE */
    EECR |= (1<EEERE);

    /* Return data from Data Register */
    return EEDR;
}
```

Note: 1. See “About Code Examples” on page 11.

### Preventing EEPROM Corruption

7.3.2 Preventing EEPROM Corruption

During periods of low $V_{CC}$ the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low $V_{CC}$ reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.
7.4 I/O Memory

The I/O space definition of the ATmega640/1280/1281/2560/2561 is shown in “Register Summary” on page 410.

All ATmega640/1280/1281/2560/2561 I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega640/1280/1281/2560/2561 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0x1FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

The I/O and peripherals control registers are explained in later sections.

7.4.1 General Purpose I/O Registers

The ATmega640/1280/1281/2560/2561 contains three General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions. See “Register Description” on page 35.
8. External Memory Interface

With all the features the External Memory Interface provides, it is well suited to operate as an interface to memory devices such as External SRAM and Flash, and peripherals such as LCD-display, A/D, and D/A. The main features are:

- Four different wait-state settings (including no wait-state)
- Independent wait-state setting for different External Memory sectors (configurable sector size)
- The number of bits dedicated to address high byte is selectable
- Bus keepers on data lines to minimize current consumption (optional)

8.1 Overview

When the eXternal MEMory (XMEM) is enabled, address space outside the internal SRAM becomes available using the dedicated External Memory pins (see Figure 1-3 on page 4, Table 12-3 on page 78, Table 12-9 on page 82, and Table 12-21 on page 90). The memory configuration is shown in Figure 8-1.

Figure 8-1. External Memory with Sector Select
8.1.1 Using the External Memory Interface

The interface consists of:

- AD7:0: Multiplexed low-order address bus and data bus.
  - A15:8: High-order address bus (configurable number of bits).
- ALE: Address latch enable.
- RD: Read strobe.
- WR: Write strobe.

The control bits for the External Memory Interface are located in two registers, the External Memory Control Register A – XMCRA, and the External Memory Control Register B – XMCRB.

When the XMEM interface is enabled, the XMEM interface will override the setting in the data direction registers that corresponds to the ports dedicated to the XMEM interface. For details about the port override, see the alternate functions in section “I/O-Ports” on page 70. The XMEM interface will auto-detect whether an access is internal or external. If the access is external, the XMEM interface will output address, data, and the control signals on the ports according to Figure 8-3 on page 31 (this figure shows the wave forms without wait-states). When ALE goes from high-to-low, there is a valid address on AD7:0. ALE is low during a data transfer. When the XMEM interface is enabled, also an internal access will cause activity on address, data and ALE ports, but the RD and WR strobes will not toggle during internal access. When the External Memory Interface is disabled, the normal pin and data direction settings are used. Note that when the XMEM interface is disabled, the address space above the internal SRAM boundary is not mapped into the internal SRAM. Figure 8-2 on page 30 illustrates how to connect an external SRAM to the AVR using an octal latch (typically “74 × 573” or equivalent) which is transparent when G is high.

8.1.2 AddressLatch Requirements

Due to the high-speed operation of the XRAM interface, the address latch must be selected with care for system frequencies above 8 MHz @ 4V and 4 MHz @ 2.7V. When operating at conditions above these frequencies, the typical old style 74HC series latch becomes inadequate. The External Memory Interface is designed in compliance to the 74AHC series latch. However, most latches can be used as long they comply with the main timing parameters. The main parameters for the address latch are:

- D to Q propagation delay (t_{PD}).
- Data setup time before G low (t_{SU}).
- Data (address) hold time after G low (t_{TH}).

The External Memory Interface is designed to guaranty minimum address hold time after G is asserted low of t_{H} = 5 ns. Refer to t_{LAXX} / t_{LLAXX} ST in “External Data Memory Timing” Tables 30-9 through Tables 30-16 on pages 378 - 381. The D-to-Q propagation delay (t_{PD}) must be taken into consideration when calculating the access time requirement of the external component. The data setup time before G low (t_{SU}) must not exceed address valid to ALE low (t_{AVULLC}) minus PCB wiring delay (dependent on the capacitive load).
8.1.3 Pull-up and Bus-keeper

The pull-ups on the AD7:0 ports may be activated if the corresponding Port register is written to one. To reduce power consumption in sleep mode, it is recommended to disable the pull-ups by writing the Port register to zero before entering sleep.

The XMEM interface also provides a bus-keeper on the AD7:0 lines. The bus-keeper can be disabled and enabled in software as described in “XMCRC – External Memory Control Register C” on page 38. When enabled, the bus-keeper will keep the previous value on the AD7:0 bus while these lines are tri-stated by the XMEM interface.

8.1.4 Timing

External Memory devices have different timing requirements. To meet these requirements, the XMEM interface provides four different wait-states as shown in Table 8-3 on page 38. It is important to consider the timing specification of the External Memory device before selecting the wait-state. The most important parameters are the access time for the external memory compared to the set-up requirement. The access time for the External Memory is defined to be the time from receiving the chip select/address until the data of this address actually is driven on the bus. The access time cannot exceed the time from the ALE pulse must be asserted low until data is stable during a read sequence (See $t_{DVRH}$ in Tables 30-9 through Tables 30-16 on pages 378 - 381). The different wait-states are set up in software. As an additional feature, it is possible to divide the external memory space in two sectors with individual wait-state settings. This makes it possible to connect two different memory devices with different timing requirements to the same XMEM interface. For XMEM interface timing details, please refer to Table 30-9 on page 378 to Table 30-16 on page 381 and Figure 30-9 on page 381 to Figure 30-12 on page 383 in the “External Data Memory Timing” on page 378.

Note that the XMEM interface is asynchronous and that the waveforms in the following figures are related to the internal system clock. The skew between the internal and external clock (XTAL1) is not guaranteed (varies between devices temperature, and supply voltage). Consequently, the XMEM interface is not suited for synchronous operation.
Figure 8-3. External Data Memory Cycles without Wait-state (SRWn1=0 and SRWn0=0)\(^{(1)}\)

System Clock (CLK\(_{CPU}\))

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALE</td>
<td>A15:8 Prev. addr.</td>
<td>Address</td>
<td>Address</td>
</tr>
<tr>
<td>DA7:0 Prev. data</td>
<td>Address</td>
<td>DA7:0 Prev. data</td>
<td>Address</td>
</tr>
<tr>
<td>WR</td>
<td>WR</td>
<td>WR</td>
<td>WR</td>
</tr>
<tr>
<td>DA7:0 (XMBK = 0) Prev. data</td>
<td>Address</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>DA7:0 (XMBK = 1) Prev. data</td>
<td>Address</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>RD</td>
<td>RD</td>
<td>RD</td>
<td>RD</td>
</tr>
</tbody>
</table>

Read

Note: 1. SRWn1 = SRW11 (upper sector) or SRW01 (lower sector), SRWn0 = SRW10 (upper sector) or SRW00 (lower sector). The ALE pulse in period T4 is only present if the next instruction accesses the RAM (internal or external).

Figure 8-4. External Data Memory Cycles with SRWn1 = 0 and SRWn0 = 1\(^{(1)}\)

System Clock (CLK\(_{CPU}\))

<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALE</td>
<td>A15:8 Prev. addr.</td>
<td>Address</td>
<td>Address</td>
<td>Address</td>
</tr>
<tr>
<td>DA7:0 Prev. data</td>
<td>Address</td>
<td>DA7:0 Prev. data</td>
<td>Address</td>
<td>Address</td>
</tr>
<tr>
<td>WR</td>
<td>WR</td>
<td>WR</td>
<td>WR</td>
<td>WR</td>
</tr>
<tr>
<td>DA7:0 (XMBK = 0) Prev. data</td>
<td>Address</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>DA7:0 (XMBK = 1) Prev. data</td>
<td>Address</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>RD</td>
<td>RD</td>
<td>RD</td>
<td>RD</td>
<td>RD</td>
</tr>
</tbody>
</table>

Read

Note: 1. SRWn1 = SRW11 (upper sector) or SRW01 (lower sector), SRWn0 = SRW10 (upper sector) or SRW00 (lower sector). The ALE pulse in period T5 is only present if the next instruction accesses the RAM (internal or external).
8.1.5 Using all Locations of External Memory Smaller than 64 Kbytes

Since the external memory is mapped after the internal memory as shown in Figure 8-1 on page 28, the external memory is not addressed when addressing the first 8,704 bytes of data space. It may appear that the first 8,704 bytes of the external memory are inaccessible (external memory addresses 0x0000 to 0x21FF). However, when connecting an external memory smaller than 64 Kbytes, for example 32 Kbytes, these locations are easily accessed simply by addressing from address 0x8000 to 0xA1FF. Since the External Memory Address bit A15 is not connected to the external memory, addresses 0x8000 to 0xA1FF will appear as addresses 0x0000 to 0x21FF for the external memory. Addressing above address 0xA1FF is not recommended, since this will address an external memory location that is already accessed by another (lower) address. To
the Application software, the external 32 Kbytes memory will appear as one linear 32 Kbytes address space from 0x2200 to 0xA1FF. This is illustrated in Figure 8-7.

Figure 8-7. Address Map with 32 Kbytes External Memory

8.1.6 Using all 64 Kbytes Locations of External Memory

Since the External Memory is mapped after the Internal Memory as shown in Figure 8-1 on page 28, only 56 Kbytes of External Memory is available by default (address space 0x0000 to 0x21FF is reserved for internal memory). However, it is possible to take advantage of the entire External Memory by masking the higher address bits to zero. This can be done by using the XMMn bits and control by software the most significant bits of the address. By setting Port C to output 0x00, and releasing the most significant bits for normal Port Pin operation, the Memory Interface will address 0x0000 - 0x2FFF. See the following code examples.

Care must be exercised using this option as most of the memory is masked away.
Assembly Code Example

```assembly
; OFFSET is defined to 0x4000 to ensure external memory access
; Configure Port C (address high byte) to output 0x00 when the pins are released
; for normal Port Pin operation
ldi r16, 0xFF
out DDRC, r16
ldi r16, 0x00
out PORTC, r16
; release PC7:6
ldi r16, (1<<XMM1)
sts XMCRB, r16
; write 0xAA to address 0x0001 of external memory
ldi r16, 0xaa
sts 0x0001+OFFSET, r16
; re-enable PC7:6 for external memory
ldi r16, (0<<XMM1)
sts XMCRB, r16
; store 0x55 to address (OFFSET + 1) of external memory
ldi r16, 0x55
sts 0x0001+OFFSET, r16
```

C Code Example

```c
#define OFFSET 0x4000

void XRAM_example(void)
{
    unsigned char *p = (unsigned char *) (OFFSET + 1);

    DDRC = 0xFF;
    PORTC = 0x00;

    XMCRB = (1<<XMM1);

    *p = 0xaa;

    XMCRB = 0x00;

    *p = 0x55;
}
```

Note: 1. See “About Code Examples” on page 11.
8.2 Register Description

8.2.1 EEPROM registers

8.2.1.1 EEARH and EEARL – The EEPROM Address Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x22 (0x42)</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>EEAR11</td>
<td>EEAR10</td>
<td>EEAR9</td>
<td>EEAR8</td>
</tr>
<tr>
<td>0x21 (0x41)</td>
<td>EEAR7</td>
<td>EEAR6</td>
<td>EEAR5</td>
<td>EEAR4</td>
<td>EEAR3</td>
<td>EEAR2</td>
<td>EEAR1</td>
<td>EEAR0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read/Write</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- Bits 15:12 – Res: Reserved Bits
  These bits are reserved bits and will always read as zero.

- Bits 11:0 – EEAR8:0: EEPROM Address
  The EEPROM Address Registers – EEARH and EEARL specify the EEPROM address in the 4 Kbytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 4096. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

8.2.1.2 EEDR – The EEPROM Data Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20 (0x40)</td>
<td>MSB</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bits 7:0 – EEDR7:0: EEPROM Data
  For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

8.2.1.3 EECR – The EEPROM Control Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1F (0x3F)</td>
<td>–</td>
<td>–</td>
<td>EEPM1</td>
<td>EEM0</td>
<td>EERIE</td>
<td>EEMPE</td>
<td>EEPE</td>
<td>EEPE</td>
</tr>
<tr>
<td>Read/Write</td>
<td>R</td>
<td>R</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
<td>R/W</td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bits 7:6 – Res: Reserved Bits
  These bits are reserved bits and will always read as zero.

- Bits 5, 4 – EEPM1 and EEPM0: EEPROM Programming Mode Bits
  The EEPROM Programming mode bit setting defines which programming action that will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the Erase and Write operations in two different operations. The Programming times for the different modes are shown in Table 8-1 on page 36. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.
Table 8-1. EEPROM Mode Bits

<table>
<thead>
<tr>
<th>EEPM1</th>
<th>EEPM0</th>
<th>Programming Time</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3.4 ms</td>
<td>Erase and Write in one operation (Atomic Operation)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1.8 ms</td>
<td>Erase Only</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1.8 ms</td>
<td>Write Only</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>–</td>
<td>Reserved for future use</td>
</tr>
</tbody>
</table>

- **Bit 3 – EERIE: EEPROM Ready Interrupt Enable**
  Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEPE is cleared.

- **Bit 2 – EEMPE: EEPROM Master Programming Enable**
The EEMPE bit determines whether setting EEPE to one causes the EEPROM to be written. When EEMPE is set, setting EEPE within four clock cycles will write data to the EEPROM at the selected address. If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to one by software, hardware clears the bit to zero after four clock cycles. See the description of the EEPE bit for an EEPROM write procedure.

- **Bit 1 – EEPE: EEPROM Programming Enable**
The EEPROM Write Enable Signal EEPE is the write strobe to the EEPROM. When address and data are correctly set up, the EEPE bit must be written to one to write the value into the EEPROM. The EEPE bit must be written to one before a logical one is written to EEPE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

  1. Wait until EEPE becomes zero.
  2. Wait until SPMEN in SPMCSR becomes zero.
  3. Write new EEPROM address to EEAR (optional).
  4. Write new EEPROM data to EEDR (optional).
  5. Write a logical one to the EEMPE bit while writing a zero to EEPE in EECR.
  6. Within four clock cycles after setting EEMPE, write a logical one to EEPE.

  The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted. See “Memory Programming” on page 335 for details about Boot programming.

  **Caution:** An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

  When the write access time has elapsed, the EEPE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.