University of Florida
Department of Electrical and Computer Engineering
EEL5666
Intelligent Machines Design Laboratory

Final Report

Lil' Helper

By: Jennifer Labush

Table of Contents

Abstract

The following paper describes the deigsn of an autonomous mobile robot that vacuums a room. The robot will after being turned on travel around the house without user interventention. It will avoid obstacles so that is will not get stuck if there is a table or other object in the room. The motivation for this project is just avoid the monotonous tacsk of vacuuming.

Executive Summary

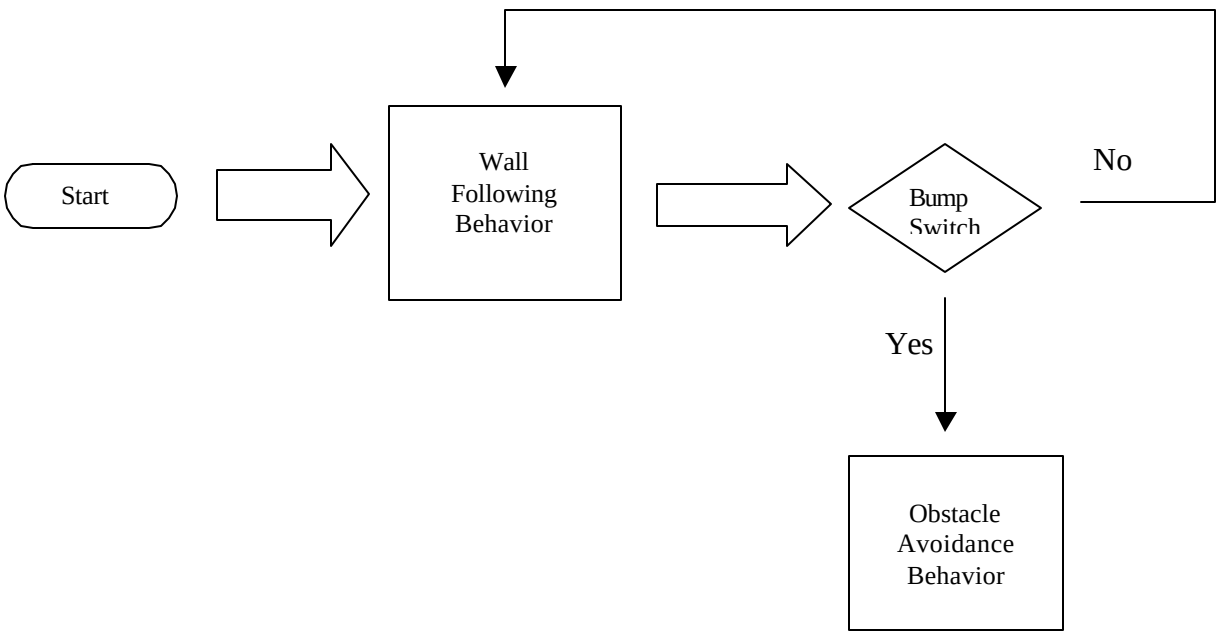
Lil' Helper is a mobile autonomous robot whose sole purpose is to vacuum. It avoids obstacles in the room and vaccums as it moves around.

Lil' Helper uses it IR dectectors to avoid obstacles while vacuuming. IT also implements wall following to more thoroughly clean the floor. The vacumming is done by a Dust Buster attached to the platform.

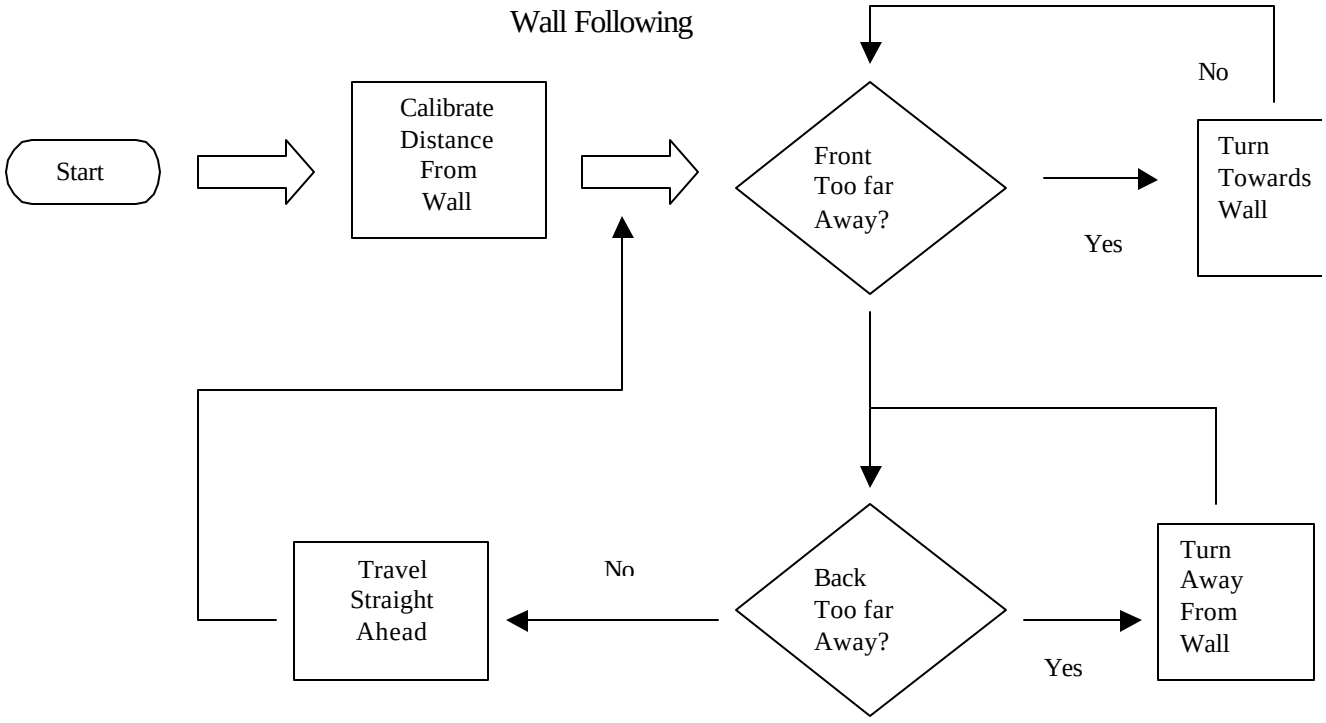
Integrated System

Lil' Helper is an integration of sensors, microprocessor, platform and a dust buster. The robot uses its sensors to react to its environment. It has two 8-packs of batteries one for the dust buster and the other for the electronics. The Servos use 6 batteries tapped off the 8-pack for the electronics.

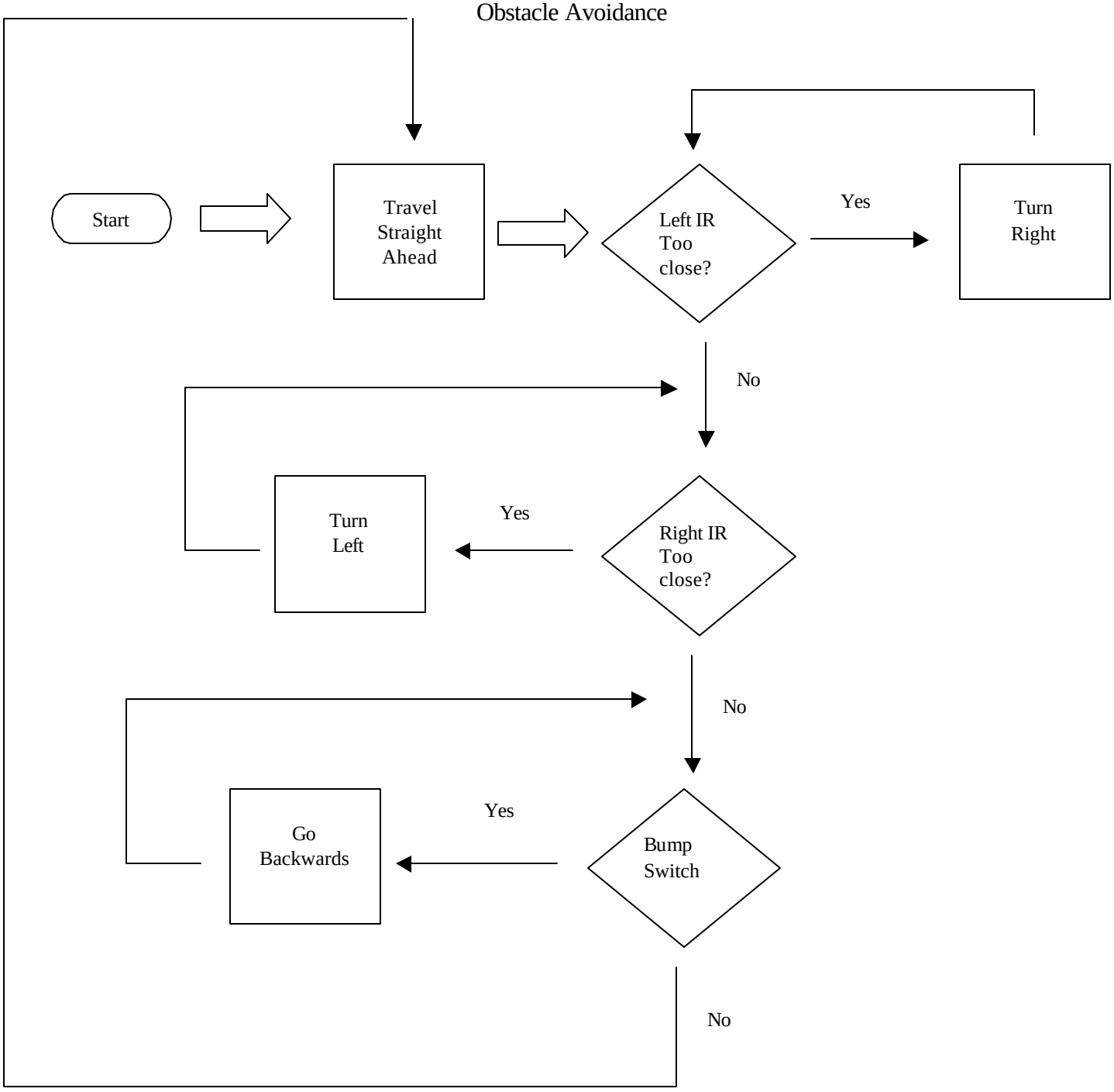
Overall View



Wall Following



Obstacle Avoidance



Mobile Platform

Lil' Helper has a wooden platform with a Dust Buster attached. The dimensions are approximately 18 inches long, 9 inches wide, and 7 inches tall. The robot has three-inch wheels.

The wooden platform was designed in AutoCAD and is 11 inches long, 6 inches wide and, 4.25 inches tall. It consists of ten pieces: six pierces for holding on the servos, bottom, a top and two sides. The microprocessor board is attached to the underside of the top of the robot. The base holds everything else. The IR's and Dust Buster are attached with heavy duty Velcro. The batteries and remainder of electronics rest on the base.

Actuation

There are a total of three actuators on Lil' Helper. These include two partially hacked Servos for drive motors. There is also a Dust Buster motor, which does the actual vacuuming. These three actuators allow the robot to move around and vacuum the carpet.

The servos used are Cirrus CS-80MG Buggy Servos ordered from Hobby People (See Appendix). Servo specs:

Size (inches)
1.60 x 1.49 x 0.79

Weight
2.01oz (57g)

Speed (6v)
.25 sec/60 deg

Torque (6v)
129.8 oz/in (9.4 Kg/cm)

Output
Ball Bearings

The Dust Buster Motor is controlled by the HC11 using a MOSFET circuit to turn it on and off. It is powered by an 8-pack of batteries.

Sensors

Sensors are useful for providing feedback to the robot. Lil' Helper has a total of 5 sensors. There are two different types of sensors Infrared Detectors and Bump Switches. These sensors provide information to the robot about it surroundings allowing it to avoid obstacles and follow the wall.

The IR sensors used are Sharp GP2D 12 14. There are a total of 4 of these. There are two IR's in front for obstacle avoidance. There are also two on the side for wall following.

Behaviors

Lil' Helper has three main behaviors which help it accomplish the goal of cleaning the floor. These behaviors are Obstacle Avoidance, Wall Following, and Vacuuming.

The Obstacle Avoidance behavior causes the robot to turn when it is near an obstacle to avoid hitting it. The Wall Following behavior was implemented because presumably there is a lot of dirt near walls. This would be missed if only obstacle avoidance was implemented because the robot would turn before getting to the wall to avoid the "obstacle".

The vacuum turns on when the robot is turned on. It is turned off when the robot is turning to conserve battery power and extend the life of the batteries. These behaviors combine to vacuum a room completely.

Conclusion

The design has gone through many revisions to bring it to its current state. It does what it set out to do with the exception of the bag full sensor. I originally intended to implement this sensor but due to time restrictions I did not complete it.

Lil' Helper is a useful robot for cleaning your house. It works well even though slowly. The main downfall of the robot is battery consumption. The batteries would have to be changed several times in vacuuming a room. In the future I would like to find a better longer lasting battery source. In conclusion, Lil' Helper could clean your house for you pretty well, as long as you had a few hours to wait!

Appendix

//

```
// Jennifer Labush
////////////////////////////////////////////////

#include <stdio.h>
#include <mil.h>
#include <hc11.h>

   extern void _start(void);

#define DUMMY_ENTRY (void (*)(void))0xFFFF
#define LEFT 0
#define RIGHT 1
#define PERIOD 30000
#define LEFT_IR analog[0]


#define STOP 0
#define SFORWARD 1
#define SREVERSE 3
#define SRIGHT 4
#define SLEFT 5
#define FRIGHT 6
#define FFORWARD 2

#pragma interrupt_handler TOC1_isr, TOC2_isr, TOC3_isr, TOC4_isr,
TOC5_isr;

void init_pwm(void);
void init_var(void);
void TOC1_isr(void);
void TOC2_isr(void);
void TOC3_isr(void);
void wait(void);
 void updatesensor(void);
 void init_ad(void);
void robot_move(int);

int dutyLEFT;
int dutyRIGHT;
int basedutyLEFT;
int basedutyRIGHT;
int baseFRONT;
int baseBACK;
int time;
int i;
int j;
int sensordata[8];
char clear[]= "\x1b\x5B\x32\x4A\x04"; /*clear screen*/
char place[]= "\x1b[1;1H";          /* Home cursor */
int loop=1;
int loop2=1;
int choice;



void main(void)
{
```

```
 printf("%s", clear);  /*clear screen*/
  printf("%s", place);  /*home cursor*/
printf("it's working here ");


 init_pwm();

  init_ad();

   basedutyLEFT=3000+110;
   basedutyRIGHT=3000+170;



    //begin wall following
updatesensor();
//front is sensordata[5]
//back is sensordata[6]
baseFRONT= sensordata[5];
baseBACK= sensordata[6];


while(loop)
{
updatesensor();
robot_move(SFORWARD);
while ((sensordata[5] > baseFRONT+3) && (sensordata[6] < baseBACK-3))
{ robot_move(SLEFT);
updatesensor();
}
robot_move(SFORWARD);

while ((sensordata[6] > baseBACK+3) && (sensordata[5] < baseFRONT-3))
{ robot_move(SRIGHT);
updatesensor();
}
robot_move(SFORWARD);

if (sensordata[0]<30)
loop=0;

}


    //end wall following


/*

while (1)
{
 //STOP
 //SLOW FORWARD
 //TURN RIGHT
 //SLOW REVERSE
 //TURN LEFT
```

```
 //TURN RIGHT
 //FAST FORWARD
 printf("1: Stop");
 printf("2: Slow Forward");
 printf("3: slow revers");
  printf("4: turn right slow");
   printf("5: turn left");
    printf("6: turn right fast");
     printf("7: FAst forward \n");

choice=getchar();
//if (choice=='1')
//robot_move(0);
//if (choice=='2')
robot_move(SFORWARD);
printf("here you are");
//robot_move(choice);


}
*/


//printf("just before the wait");
//getchar();
  time=500;
   wait();
   printf ("vacuum on");
   SET_BIT(PORTA, 0x10);
   //time=600;
   //wait();
  // printf("vacuum off");
//      CLEAR_BIT(PORTA, 0x10);
   //   choice='7';
//  robot_move(2);
robot_move(SFORWARD);
   while (1)
   {


   updatesensor();
   //sensordata[3] is left IR
   //sensordata[1] is right IR
   //sensordata[6] is bump switch
      if (sensordata[1] >85 && sensordata[7] >85)
      {
      //goes back for a little while
//printf ("should start going back here");
robot_move(SREVERSE);

   time=500;
   wait();
robot_move(FFORWARD);
   }
   else if (sensordata[7] > 85 )
   {
   //turn left
```

```
   printf ("turning left");
   printf("vacuum off");
      CLEAR_BIT(PORTA, 0x10);
robot_move(SLEFT);
   time=500;
   wait();
robot_move(FFORWARD);
printf ("vacuum on");
   SET_BIT(PORTA, 0x10);

   }
   else if (sensordata[1] > 85)
   {
   //turn right
   printf("vacuum off");
      CLEAR_BIT(PORTA, 0x10);

   printf("turning right");
robot_move(SRIGHT);
   time=500;
   wait();
robot_move(FFORWARD);
printf ("vacuum on");
   SET_BIT(PORTA, 0x10);

}
//printf ("should go back if %d >30",sensordata[0]);

if (sensordata[0]<30)
{
//bump switch activated so go back
printf ("the robot should go back now!");
robot_move(SREVERSE);
time=300;
wait();
robot_move(FFORWARD);



   }

   }
//   }


//printf ("the duty cycle is %d", dutyLEFT);

}

 void init_pwm(void){
      INTR_OFF();

      SET_BIT(TMSK1, 0x80);         //Enable OC1
      SET_BIT(OC1M, 0x60);          //OC1 control
      SET_BIT(OC1D, 0x60);
```

```
      SET_BIT(TMSK1, 0x40);         //Enable OC2
      SET_BIT(TCTL1, 0x80);         //Clear on OC
      CLEAR_BIT(TCTL1, 0x40);

      SET_BIT(TMSK1, 0x20);         //Enable OC3
      SET_BIT(TCTL1, 0x20);         //Clear on OC
      CLEAR_BIT(TCTL1, 0x10);


      INTR_ON();

   }//End init_pwm

  void TOC1_isr(void){

      CLEAR_FLAG(TFLG1, 0x80);

    TOC2 = TOC1 + dutyLEFT;
    TOC3 = TOC1 + dutyRIGHT;
     TOC1 += PERIOD;


//   printf ("oc1\n");
   }//End TOC1

    void TOC2_isr(void){

      CLEAR_FLAG(TFLG1, 0x40);

   }//End TOC2

    void TOC3_isr(void){

      CLEAR_FLAG(TFLG1, 0x20);

   }//End TOC3



  void wait(void){

      for (i = time; i > 0; i--)
           { for (j = time; j > 0; j--);}
   }

  void init_ad(void)
  {

      SET_BIT(OPTION, 0x80);
      time = 100;
      wait();
   }


void updatesensor(void){

      ADCTL = 0x10;
```

```
      time = 12;
      wait();

      sensordata[0] = ADR1;
      sensordata[1] = ADR2;
      sensordata[2] = ADR3;
      sensordata[3] = ADR4;

      ADCTL = 0x14;

      time = 12;
      wait();

      sensordata[4] = ADR1;
      sensordata[5] = ADR2;
      sensordata[6] = ADR3;
      sensordata[7] = ADR4;

       printf("%s", clear);  /*clear screen*/
  printf("%s", place);  /*home cursor*/
printf ("here I came to save the day\n");
      printf("Sensor 0: %d, ", sensordata[0]);
      printf("Sensor 1: %d, ", sensordata[1]);
      printf("Sensor 2: %d, ", sensordata[2]);
      printf("Sensor 3: %d,\n", sensordata[3]);
      printf("Sensor 4: %d, ", sensordata[4]);
      printf("Sensor 5: %d, ", sensordata[5]);
      printf("Sensor 6: %d, ", sensordata[6]);
      printf("Sensor 7: %d,\n", sensordata[7]);


      //printf("Press any key to continue\n");
      //getchar();
   }

void robot_move( int choice2)
{           //begin robot_move


   if (choice2 ==0)
   {
   //STOP ROBOT
   dutyLEFT= basedutyLEFT;
   dutyRIGHT=basedutyRIGHT;

   }
   if (choice2 ==1)
   {
   printf("the robot should be going forward slowly");
   //SLOW FORWARD
   dutyLEFT= basedutyLEFT-100;
   dutyRIGHT=basedutyRIGHT+100;
   }

   if (choice2 ==2)
   {
```

```
   //FAST FORWARD
   dutyLEFT= basedutyLEFT-200;
   dutyRIGHT=basedutyRIGHT+200;
   }

   if (choice2 ==3)
   {
   //SLOW REVERSE
   dutyLEFT= basedutyLEFT+200;
   dutyRIGHT=basedutyRIGHT-200;
   }


   if (choice2 ==4)
   {
   //TURN LEFT SLOW
   dutyLEFT= basedutyLEFT+100;
   dutyRIGHT=basedutyRIGHT+200;
   }

   if (choice2==5)
   {
   dutyLEFT= basedutyLEFT-100;
   dutyRIGHT=basedutyRIGHT-200;
   }


      if (choice2 ==6)
   {
   //TURN LEFT FAST
   dutyLEFT= basedutyLEFT+200;
   dutyRIGHT=basedutyRIGHT+100;
   }


}     //end robot_move


#pragma abs_address:0xffd6
/* change the above address if your vector starts elsewhere */

   void (*interrupt_vectors[])(void) =
   {
   /* to cast a constant, say 0xb600, use(void (*)())0xb600 */

   DUMMY_ENTRY,   /* SCI, RS232 protocol */
   DUMMY_ENTRY,   /* SPI, high speed synchronous serial*/
   DUMMY_ENTRY,   /* Pulse accumulator input edge */
   DUMMY_ENTRY,   /* Pulse accumulator overflow */
   DUMMY_ENTRY,   /* Timer overflow */
   DUMMY_ENTRY,         /* TOC5 */
   DUMMY_ENTRY,         /* TOC4 */
   TOC3_isr,            /* TOC3 */
   TOC2_isr,            /* TOC2 */
   TOC1_isr,            /* TOC1 */
   DUMMY_ENTRY,   /* TIC3 */
   DUMMY_ENTRY,   /* TIC2 */
```

```
   DUMMY_ENTRY,   /* TIC1 */
   DUMMY_ENTRY,   /* RTI */
   DUMMY_ENTRY,   /* IRQ */
   DUMMY_ENTRY,   /* XIRQ */
   DUMMY_ENTRY,   /* SWI */
   DUMMY_ENTRY,   /* ILLOP */
   DUMMY_ENTRY,   /* COP */
   DUMMY_ENTRY,   /* CLMON */
   _start       /* RESET */
   };

#pragma end_abs_address
```