Making a Code Library
Ralph Alvarez
Dewitt Perry Middle School
culpd@cfbisd.edu

Making a Code Library

Introduction

What makes your robot move? Programming, right? Different actions your robot makes are made by using different commands. Robots we build for Botball almost always use the same commands and functions, why type the same things over and over again every time you build a new robot? One thing you could do is to make a code library. A code library would help you remember the basic commands and help your robotics team become more organized when it comes to programming.   This paper will present a very basic code library for use in Botball.

Needed Functions

The first thing you should do when your team sets out to make a code library is to sit down and determine what functions you want in your library. I will concentrate on only movement functions. Mobile robots require the basic maneuvering commands such as forward, backward, left, and right. These commands are described by speed, ranging from -1000 to 1000, and the position, which are counted by “clicks”.

```
#use "botball.ic"

#define RM 0

#define LM 2

#define LIGHT 3
```

The first thing I have done is use some #define statements to define our motors and sensors. Above, the “RM/LM” represents the motor; LM represents the left motor and RM represents the right motor. The “LIGHT” represents the light sensor that is required to sense the light from the light bulb on the Botball table. The numbers represents the slot the sensor/motor is plugged into. These numbers can be easily changed to fit the current robot.

The basic movement functions appear below:

```
void forward(int speed,long pos)

{

   mrp(RM,speed,pos);

   mrp(LM,speed,pos);

   bmd(LM);

}


void back(int speed,long pos)

{

   mrp(RM,speed,-pos);

   mrp(LM,speed,-pos);

   bmd(LM);

}


void left(int speed, long pos)

{

   mrp(RM,speed,pos);

   mrp(LM,speed,-pos);

   bmd(LM);

}
```

```
void right(int speed, long pos)

{

   mrp(RM,speed,-pos);

   mrp(LM,speed,pos);

   bmd(LM);

}
```

Above, the four basic movement commands: left, right, forward, and backward. The "mrp" in the commands represents how far you want the motor to move, which is then followed by one of the motors and the speed and position. The "bmd" represents block motor done, which will wait until the motor is done moving before returning from the function.

```
void our_freeze()

{

   clear_motor_position_counter(RM);

   clear_motor_position_counter(LM);

   while (1)

     {

      if(get_motor_position_counter(RM)!=0L)

        {

         mtp(RM,500,0L);

      }

      if(get_motor_position_counter(LM)!=0L)

        {

         mtp(LM,500,0L);

      }

   }

}
```

The above function is one that can stop a wheel on a robot and freeze it in position. We put this in our library because we found many situations when we wanted our wheels to stop and not move, especially when you wanted traction. At the beginning of the function, the motor positions are to be cleared. It then enters a loop that states that if the motor positions do not read 0, the motor will move to the 0 position. We use this as a function in the background so we can start and stop it whenever we need to.

Conclusion

A code library could help a team keep on the same page and prevent anyone from being confused. In addition it saves lots of time in programming because you do not have to keep writing the same functions over and over again. It would help different teams understand the programs made by each other. These are just some of the functions we have in our library, you should make your own library and use functions you think would be useful to your team. Altogether, a code library could keep a team organized and help prevent confusion.