AUTOFLEX 2.0

New and Improved Autonomous Programming Tool for FIRST Robots

by Brian Cieslak

In the March '06 issue of *SERVO Magazine*, I introduced you to a program called AutoFlex — a tool used for developing autonomous routines for FRC (First Robotics Competition) robot controllers.

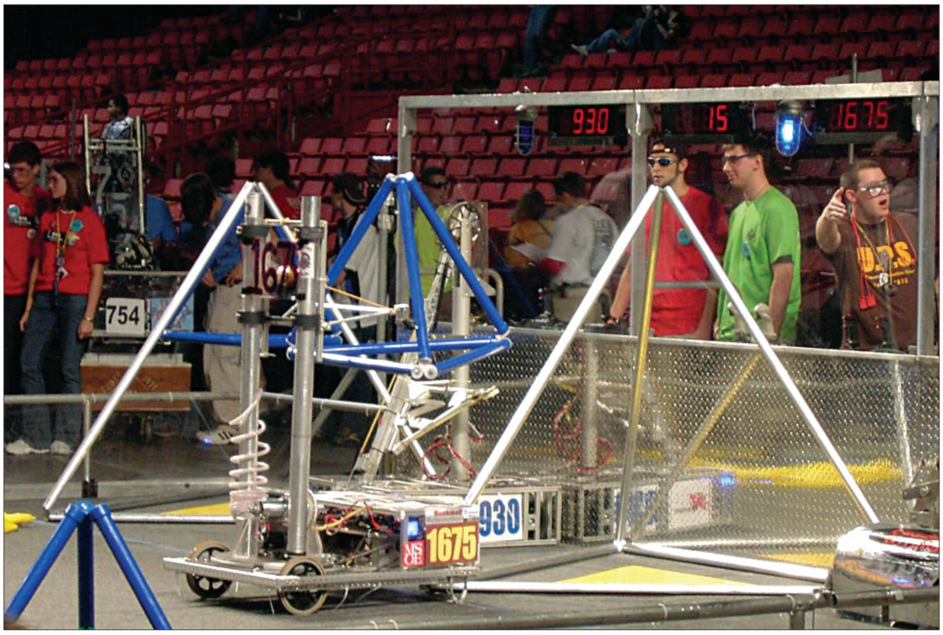


FIGURE 1. Team 1675's first robot programmed with Autoflex for the FIRST Triple Play competition would know one tetra from the goal and cap the second during the autonomous period.

The program was created by members of FRC Team 1675 when they realized that they were going to the FIRST (For Inspiration and Recognition of Science and Technology) National Championships in Atlanta, GA without any autonomous functionality for their robot. Without having access to the robot until the event, they needed a way to quickly program the robot to perform some task during the autonomous period. The solution was to create a program that would allow the team's driver to teach the robot what it had to do during the autonomous period by recording the driver's commands as he drove through the autonomous routine. Training took place on the practice field before the matches started. At the beginning of the match, the robot would repeat the commands that it was taught.

In 2005 during the Triple play competition, the robot scored two tetras during each autonomous period. During the 2006 Aim High competition, the robot could drive up to the goal and shoot an entire magazine of balls through the hole (most of the time).

Photos courtesy of FRC Team 1675 — The Ultimate Protection Squad.

The original program was a little cumbersome and complicated to use. While the driver commanded the robot through the routine that was to be recorded and then played it back during the autonomous mode, a programmer — with a laptop connected to the robot via a serial cable — chased (or was chased by) the robot as he captured data. The data then had to be loaded into a file and the whole program was recompiled and reloaded into the robot. From the sidelines, this was fun to watch, but those actually involved in the process were often quite stressed and in peril.

Autoflex has been simplified and updated to version 2.0 to take advantage of the internal EEPROM memory available in the FRC robot controller. Commands are now written directly to the EEPROM memory. No more laptops and cables, editing data, and reprogramming. Programmer stress levels have been greatly reduced!

Imagine This!

During practice, you set your robot on the playing field, click a button on the operator interface and start driving. Then you set your robot back to the starting point, connect a dongle to the competition port of the operator interface and flip the dongle switch to autonomous and the robot will replay the practice session you just recorded. Don't like what you see? Just reset the dongle switch back to the off position and just click the program button again to re-record another session until you get it right. Neat, eh?? Your robot is ready to run during the autonomous period, all in about 15 minutes.

Getting Started

A .zip file can be downloaded from the FRC Team 1675 website (see Web Links sidebar) that includes a version of the FRC default code with AutoFlex included. I/O mapping for the default program is as follows:

Joystick 1 - Y axis to PWM_1
Joystick 2 - Y axis to PWM_2
Joystick 1 - X axis to PWM_5
Joystick 2 - X axis to PWM_6

Set your robot to program mode and download the FRCAutoFlex.hex file using the IFIdownloader program available from the IFI website (**www.ifirobotics.com**).

Attach a programming dongle to the competition port of the operator interface (instructions on how to make your own are also available from the IFI website) and set the autonomous switch to the open position. You are now ready to start programming your robot for autonomous operation. The FRCAutoFlexCode.hex program records four inputs: joystick 1 x-axis, joystick 1 y-axis, joystick 2 x-axis, and joystick 2 y-axis.

Click the trigger on the port 1 joystick to start recording. You now have 15 seconds to drive through your autonomous routine. After 15 seconds, the robot stops recording commands even though it lets you keep driving.

To replay what you just recorded, 'close' the autonomous switch on the dongle. Watch out! Your robot will start to execute the code you just recorded. The robot will play 15 seconds of commands and then stop until you open the autonomous switch again.

Once you are satisfied with the autonomous routine you've recorded, place a jumper on the 'digital input 1' pins. This write protects your autonomous program from being accidentally erased if you click the trigger while driving around.

That's the basic operation. Now you are an expert.

For the Beginning FIRST Programmer

If you are just learning to program a FIRST robot, a sample project that is fully functional is included in the zip file you can download from the Team 1675 website that can serve as a template to get you started. The programming kit that comes with your robot includes a disk with the MPLAB-IDE programming environment and the C18-Complier Version 2.4, as well as the downloader program. You will need these tools to compile and download your program to the robot.

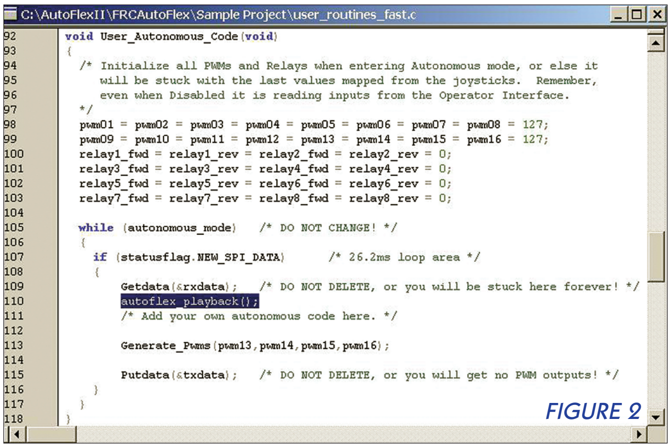
Adding Autoflex to Your Existing Code

Adding AutoFlex to your existing code is simple if all the calls to your control functions (motor control, manipulator arm, etc.) are made from the Default_Routine() function found in the User_Routines.c file.

You must do the following (refer to the sample code provided):

1) Copy the following files to your project folder, then open MPLAB and add them to your project:
   a) AutoFlex.c
   b) Autoflex.h
   c VEX_eeprom.c
   d) VEX_eeprom.h

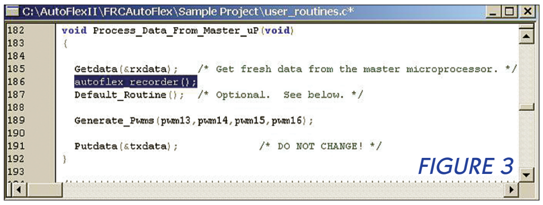
2) Open the user_routines_fast.c file. Add a call to the function autoflex_playback() to the user_autonomous_code() function as shown in Figure 2. Also add the #include"AutoFlex.h statement at the beginning of the file.


C:\AutoFlexII\FRCAutoFlex\Sample Project\user_routines_fast.c

```
void User_Autonomous_Code(void)
{
  /* Initialize all PWMs and Relays when entering Autonomous mode, or else it
     will be stuck with the last values mapped from the joysticks.  Remember,
     even when Disabled it is reading inputs from the Operator Interface.
  */
  pwm01 = pwm02 = pwm03 = pwm04 = pwm05 = pwm06 = pwm07 = pwm08 = 127;
  pwm09 = pwm10 = pwm11 = pwm12 = pwm13 = pwm14 = pwm15 = pwm16 = 127;
  relay1_fwd = relay1_rev = relay2_fwd = relay2_rev = 0;
  relay3_fwd = relay3_rev = relay4_fwd = relay4_rev = 0;
  relay5_fwd = relay5_rev = relay6_fwd = relay6_rev = 0;
  relay7_fwd = relay7_rev = relay8_fwd = relay8_rev = 0;

  while (autonomous_mode)   /* DO NOT CHANGE! */
  {
    if (statusflag.NEW_SPI_DATA)      /* 26.2ms loop area */
    {
        Getdata(&rxdata);   /* DO NOT DELETE, or you will be stuck here forever! */
        autoflex_playback();
        /* Add your own autonomous code here. */

        Generate_Pwms(pwm13,pwm14,pwm15,pwm16);

        Putdata(&txdata);   /* DO NOT DELETE, or you will get no PWM outputs! */
    }
  }
}
```

FIGURE 2

3) Open the user_routines.c file. Add a call to the function autoflex_recorder() to the Process_Data_From_ Master_ uP() function as shown in Figure 3. Also add the #include"AutoFlex.h" statement at the beginning of the file.


C:\AutoFlexII\FRCAutoFlex\Sample Project\user_routines.c*

```
void Process_Data_From_Master_uP(void)
{

  Getdata(&rxdata);   /* Get fresh data from the master microprocessor. */
  autoflex_recorder();
  Default_Routine();  /* Optional.  See below. */

  Generate_Pwms(pwm13,pwm14,pwm15,pwm16);

  Putdata(&txdata);             /* DO NOT CHANGE! */
}
```

FIGURE 3

4) Open the main.c file. Add a call to the function rewind_autoflex_playback() to the main()function as shown in Figure 5. Also add the #include"AutoFlex.h" statement at the beginning of the file.

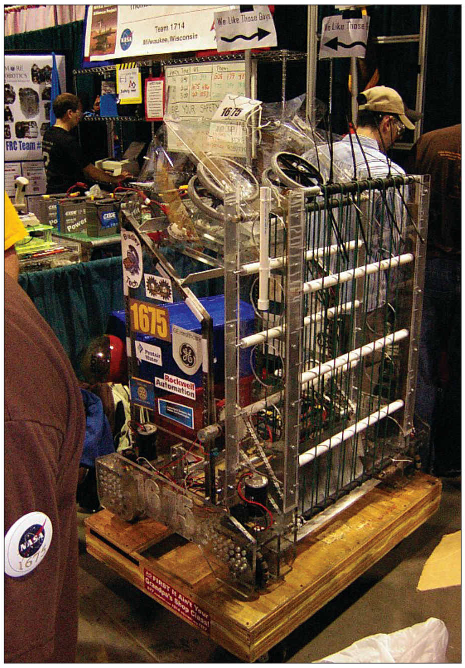
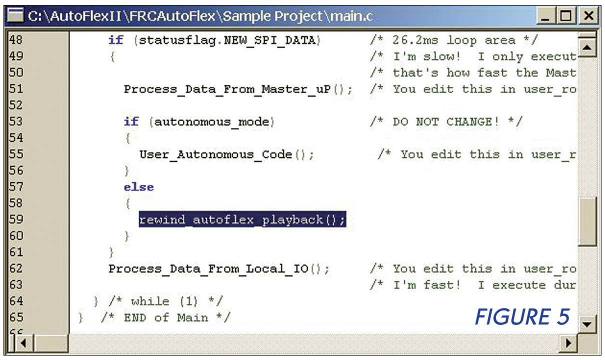


FIGURE 4. Team 1675's Aim High robot would drive up to the goal and shoot most of its 10 balls through the hole.



```
    if (statusflag.NEW_SPI_DATA)      /* 26.2ms loop area */
    {                                 /* I'm slow!  I only execut
                                      /* that's how fast the Mast
      Process_Data_From_Master_uP();  /* You edit this in user_ro

      if (autonomous_mode)            /* DO NOT CHANGE! */
      {
        User_Autonomous_Code();        /* You edit this in user_r
      }
      else
      {
        rewind_autoflex_playback();
      }
    }
    Process_Data_From_Local_IO();     /* You edit this in user_ro
                                      /* I'm fast!  I execute dur
  } /* while (1) */
}  /* END of Main */
```

FIGURE 5

Web Links

For Autoflex files
http://team1675.com/teamdownload.html

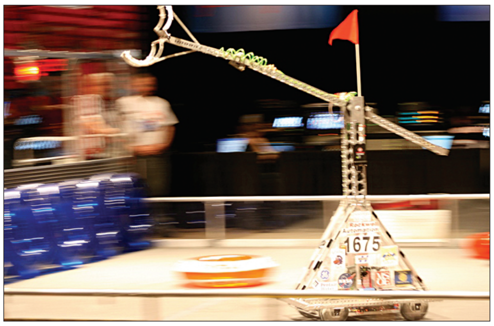
For competition port dongle
www.ifirobotics.com/oi.shtml

5) Configure Autoflex.h to reflect your robot system. Sections that you may want to consider changing include the following:

a) Determine how many inputs you want to capture and which ones.

```
//add defines here to assign
//commands to user controls that
// you want record/
#define AUTO_COMMAND1 p1_x
//left Joystick x
#define AUTO_COMMAND2 p1_y
//left Joystick y
#define AUTO_COMMAND3 p2_y
//right joystick y
#define AUTO_COMMAND4 p2_x
//right joystick x
//#define AUTO_COMMAND5
//uncomment to add another input
//#define AUTO_COMMAND6
//uncomment to add another input

// Number of inputs we plan to
//record
// Default is set up to save 4 inputs.
//You can save up to 6
// inputs. You can define two auto
//command lines above.
// then change the number of
//inputs on the line below.
#define NUM_OF_INPUTS 4i
```

FIGURE 6. Autoflex was used to program a large claw-like manipulator during the autonomous period at the beginning of the Rack-n-Roll competiton.



b) You can determine what you want to use as the 'Record Button.' The default is port 1 trigger Button.

```
//define the mechanism that will
//act as the record button.
//In this example port trigger is a
//button on the OI.
// that you would press to the
//forward position to start recording

#define \
AUTO_BUTTON_REV_THRESH      \
(unsigned char)100 // used by Vex
#define  \
AUTO_BUTTON_FWD_THRESH      \
(unsigned char)154 // used by Vex
#define \
AUTO_NEUTRAL_PWM_VALUE      \
(unsigned char)127

#define AUTO_RECORD (p1_sw_trig)
//port_1 trigger to start recording
```

c) You can adjust the length of time you want to record commands by changing the TIME_LIMIT value. Default is 150 tenths of a second (or 15 seconds). The maximum value of TIME_LIMIT depends on the number of inputs you are trying to save. The max number of command values that can be saved is 1,024. To determine the max time available, use the following formula (1024/ number_of_inputs) - 1 = max_tenths_of_seconds. For example: (1024/4 inputs)-1 = 255, so TIME_LIMIT could be set to 255 tenths_of_seconds. (25.5 seconds).

```
// The length of the autonomous
routine in tenths of seconds
#define TIME_LIMIT 150
```

d) You can assign which digital port you want to use for your WRITE_PROTECT jumper. If you don't want to write protect your autonomous code or you have used up all your digital ports, re-define WRITE_PROTECT to '1.'

```
// if jumper in place then do not
//record (assuming jumper pulls pin
//low)
#define WRITE_PROTECT \
(rc_dig_in01)
```

e) Since an FRC robot uses a longer timing interval than VEX robots during autonomous operation, uncomment the #define FRC 1 line to adjust the timing if you are adding Autoflex to a FRC robot.

No More Excuses to Sit Idle!

When I attended FIRST Regional competitions in Milwaukee, WI and Cleveland, OH and the FIRST National Championship in Atlanta, I was surprised by how many robots sat idle during the autonomous segment of the match. Our team started touting the benefits and simplicity of the Autoflex program there and enabled several teams to compete during that 15 second period at the beginning of the match. Even sending the robot out to a defensive position is better than just sitting there.

I do want to emphasize, though, that Autoflex is not a substitute for a well thought out autonomous program that uses sensors and feedback algorithms. To be truly autonomous, the robot must be aware of and react to its environment. So programmers, you are not off the hook.

See you in the 'Pits.' **SV**

Contact the Author

Brian Cieslak is a mentor for FIRST Team 1675, The Ultimate Protection Squad. He can be contacted via email at K9WIS@yahoo.com.