**CUSTOM TELEMETRY CODES**

/\*

\* Telemetry and Custom Command Sample Arduino Code

\* Telemetry Example: Monitor an extended I/O register value and send to control panel display

\* Command Example: Add Robot unique MOVE, BLINK, and SERVO commands

\*/

// include C:\Program Files (x86)\Arduino 1\_6\_5\hardware\arduino\avr\libraries

#include <Robot3DoTBoard.h> // instantiated as Robot3DoT at end of class header

#include <EEPROM.h>

#include <Wire.h> // I2C support

Robot3DoTBoard Robot3DoT; // instantiated as Robot3DoT at end of class header

/\*

\* Command Example

\* Step 1: Assign new command mnemonics ID numbers

\* In our example we will be adding 3 custom commands (2 new and one predefined).

\* The predefined MOVE command is intercepted and our robot unique code is to be

\* run instead. The MOVE command mnemonic ID 0x01 is already defined in Configure.h

\* The next two commands are new and assigned to the first two addresses in the

\* custom command address space 0x40 - 0x5F.

\*/

// modifying built6 in move handler

#define STATIC 0x40 // command

#define ROLL 0x41 // tele

#define PITCH 0x42 // tele

#define ROTARY\_R 0X43 // tele

#define ROTARY\_L 0X44 // tele

Motor motorA; // you can name your motors whatever you want

Motor motorB; // in this case I used motorA and motorB

boolean dyn;

int temp;

const uint8\_t CMD\_LIST\_SIZE = 2; // we are adding 3 commands (MOVE, BLINK, SERVO)

/\*

\* Command Example

\* Step 2: Register commands by linking IDs to their corresponding command handlers

\* In our example when the MOVE command is intercepted the moveHandler is to be run.

\* In a similar fashion the BLINK command calls the blinkHandler and SERVO the

\* servoHandler.

\*/

void moveHandler (uint8\_t cmd, uint8\_t param[], uint8\_t n);

void dynamicHandler (uint8\_t cmd, uint8\_t param[], uint8\_t n);

Robot3DoTBoard::cmdFunc\_t onCommand[CMD\_LIST\_SIZE] = {{MOVE,moveHandler}, {DYNAMIC, dynamicHandler}};

/\*

\* Telemetry Example

\* Step 1: Instantiate packet

\* In our example we simulate a current sensor wired to MOTOR 2. MOTOR2\_CURRENT\_ID

\* is defined as 0x02 in Configure.h

\* To simulate the data stream coming from the sensor we will read ATmega32U4

\* Register OCR4D which controls the duty cycle of MOTOR 2.

\*/

//Packet batteryLevel(BATTERY\_ID); // initialize the packet properties to default values //identifies the battery level of 3DoT

Packet roll(ROLL);

Packet pitch(PITCH);

//Packet shaftRight(SHAFT\_R);

//Packet shaftLeft(SHAFT\_L);

Packet rotaryRight(ROTARY\_R);

Packet rotaryLeft(ROTARY\_L);

//Packet servoAngle(SERVO\_A);

void setup()

{

Serial.begin(9600); // default = 115200

Robot3DoT.begin();

/\*

\* Command Example

\* Step 3: Tell 3DoT Robot software about new commands

\*/

Robot3DoT.setOnCommand(onCommand, CMD\_LIST\_SIZE);

/\* Telemetry Example

\* Step 2: Modify default values assigned to internal properties as needed.

\* Before a packet is created and sent, it is qualified. Specifically,

\* the data in a packet must change by some amount from the previous

\* packet and may not be sent with at a period less than some value.

\* In most cases you can leave these values at their default values.

\*/

// batteryLevel.setAccuracy(1); // change sensor accuracy from +/-2 DN to +/-1 DN (-- this line is optional --)

// batteryLevel.setSamplePeriod(500); // change sample period from 1 second to 0.5 seconds (-- this line is optional --)

// motorA.begin(5,10,9); //begin(control\_pin1,control\_pin2,pwmPin) --> set up motor control pins

// motorB.begin(19,20,6);

roll.setAccuracy(1); // change sensor accuracy from +/-2 DN to +/-1 DN (-- this line is optional --)

roll.setSamplePeriod(200); // change sample period from 1 second to 0.5 seconds (-- this line is optional --)

pitch.setAccuracy(1); // change sensor accuracy from +/-2 DN to +/-1 DN (-- this line is optional --)

pitch.setSamplePeriod(250); // change sample period from 1 second to 0.5 seconds (-- this line is optional --)

// rotaryLeft.setAccuracy(1)

// rotaryLeft.setSamplePeriod()

// rotaryRight.setAccuracy

//shaftLeft.setAccuracy(1); // change sensor accuracy from +/-2 DN to +/-1 DN (-- this line is optional --)

//shaftLeft.setSamplePeriod(100); // change sample period from 1 second to 0.5 seconds (-- this line is optional --)

//shaftRight.setAccuracy(1); // change sensor accuracy from +/-2 DN to +/-1 DN (-- this line is optional --)

//shaftRight.setSamplePeriod(100); // change sample period from 1 second to 0.5 seconds (-- this line is optional --)

dyn = 1;

temp = 0;

// servoAngle.setSamplePeriod(100); //writing the code to determine the (ignore for gifty)

}

void loop()

{

Robot3DoT.loop();

/\*

\* Telemetry Example

\* Step 3: Read sensor and send packet

\* To simulate the data stream coming from the sensor we will read ATmega32U4

\* Register OCR4D which controls the duty cycle of MOTOR 2. This code segment

\* uses the preproccessor conditional directives #if to make sure that an MCU

\* with a ATmega32U4 or ATmega16U4 is selected under Tools > Board.

\*/

// temp = batteryLevel.getVoltage();

// uint16\_t batteryReading = (uint16\_t) readFuelGauge() //battery

// batteryLevel.

uint16\_t rollReading = (uint16\_t) 35; // read 8-bit Output Compare Register Timer 4D and cast to 16-bit signed word

roll.sendSensor(rollReading);

uint16\_t pitchReading = (uint16\_t) 10; // read 8-bit Output Compare Register Timer 4D and cast to 16-bit signed word

pitch.sendSensor(pitchReading);

// uint16\_t leftReading = (uint16\_t) 100; // read 8-bit Output Compare Register Timer 4D and cast to 16-bit signed word

// shaftLeft.sendSensor(leftReading);

// uint16\_t rightReading = (uint16\_t) temp; // read 8-bit Output Compare Register Timer 4D and cast to 16-bit signed word

// shaftRight.sendSensor(rightReading);

// uint16\_t batLvl = (uint16\_t) temp; // read 8-bit Output Compare Register Timer 4D and cast to 16-bit signed word

// batteryLevel.sendSensor(batLvl);

uint16\_t leftReading = (uint16\_t) 23; //read this value to test our command

rotaryLeft.sendSensor(leftReading);

uint16\_t rightReading = (uint16\_t) 46;

rotaryRight.sendSensor(rightReading);

}

/\*

\* Command Example

\* Step 4: Write command handlers

\*/

/\*

\* User Defined Command BLINK (0x40) Example

\* A5 01 40 E4

\*/

//void MotorHandler (uint8\_t cmd, uint8\_t param[], uint8\_t n)

//{

//

//

// Serial.print("TEST");

// motorA.go(param[0],255);// go(direction,pwm) --> direction: FORWARD =1, BACKWARD =2, BRAKE = 3, RELEASE = 4; pwm range(0-255) used to control the speed of the motor

// //in my case I used 1 to move forward and a 255 for max speed

// motorB.go(param[1],255);// go(direction,pwm) --> direction: FORWARD =1, BACKWARD =2, BRAKE = 3, RELEASE = 4; pwm range(0-255) used to control the speed of the motor

//} // blinkHandler

/\*

\* Override MOVE (0x01) Command Example

\* A5 05 01 01 80 01 80 A1

\*/

void moveHandler (uint8\_t cmd, uint8\_t param[], uint8\_t n)

{

Serial.write(cmd); // move command = 0x01

Serial.write(n); // number of param = 4

/\*

\* User Defined Command SERVO (0x41) Example

\* Rotate servo to 90 degrees

\* A5 02 41 90 76

\*/

/\*

\* Static: A5 02 40 00 E7

\* Dynamic: A5 02 40 01 E6

\*/

void dynamicHandler (uint8\_t cmd, uint8\_t param[], uint8\_t n)

{

// Serial.write(cmd); // servo command = 0x41

// Serial.write(n); // number of param = 1

dyn = param[0]; // go(direction,pwm) --> direction: Dynamic =1, Static =0

temp += dyn;

//Serial.print(temp);

//Serial.print("hello");

} // servoHandler