iMicro V20 Software Guide

iMicro C Library

The iMicro C library provides the function calls for the on board motion control module and RC servo generation module.

Header File	Description	Library/Source File
MotionCtrl.h	Function call for motion control module	iMicro1.lib
RCServo.h	Function to control RC servo pulse	iMicro1.lib
CMPS03.h	Function call for CMPS03 digital compass	CMPS03.c
ADC.h	Function call to read ADC	iMicro1.lib
Serial_com.h	Interrupt driven serial input	iMicro1.lib
RF.h	RF communication routines for Radiometrix Module	RF.c
RC4620.h	iMicro macro definition	

Motion Control Functions

I2C Address Functions

`void SetI2CAddress(unsigned char addr)`	
Description	This function set the I2C address for motion controller to set command to
Parameters	addr: 8 bit address of the targeted motion controller. Note: For on-board motion controller, the I2C address is defined in the constant _IMICRO_ADDR. (_IMICRO_ADDR = 0x50)
Return Value	
Files	MotionCtrl.h, iMicro1.lib

Velocity Mode Functions

`char SetVelocity(float* pV0, float* pV1)`	
Description	This function set the speed of the two DC motors
Parameters	pV0: pointer to the floating point value of the desired velocity of Motor 0 in mm/sec. pV1: pointer to the floating point value of the desired velocity of Motor 1 in mm/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadVelocity(float* pV0, float* pV1)`	
Description	This function read the speed of the two DC motors
Parameters	pV0: pointer to the floating point value to hold current velocity of Motor 0 in mm/sec. pV1: pointer to the floating point value to hold current velocity of Motor 1 in mm/sec.
Return Value	1 if the function call is successful 0 if the function call fails

Files	MotionCtrl.h, iMicro1.lib

`char SetVelocity0(float* pV0)`	
Description	This function set the speed of the DC motor0
Parameters	pV0: pointer to the floating point value of the desired velocity of Motor 0 in mm/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadVelocity0(float* pV0)`	
Description	This function read the speed of the DC motor0
Parameters	pV0: pointer to the floating point value to hold current velocity of Motor 0 in mm/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetVelocity1(float* pV0)`	
Description	This function set the speed of the DC motor1
Parameters	pV0: pointer to the floating point value of the desired velocity of Motor 0 in mm/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadVelocity1(float* pV0)`	
Description	This function read the speed of the DC motor1
Parameters	pV0: pointer to the floating point value to hold current velocity of Motor 0 in mm/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

Position Mode Functions

`char SetPosition(float* p0, float* p1)`	
Description	This function Set the desired relative position of the two motors. The motors will run in Position Mode.
Parameters	p0: pointer to the floating point value of the desired relative position of Motor 0 in mm. p0: pointer to the floating point value of the desired relative position of Motor 1 in mm.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadPosition(float* p0, float* p1)`

Description	This function read the current absolute position of the two motor
Parameters	p0: pointer to the floating point value to hold the current absolute position of Motor 0 in mm. p1: pointer to the floating point value to hold the current absolute position of Motor 1 in mm.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetPosition0(float* p0)`	
Description	This function Set the desired relative position of the motor 0. The motors will run in Position Mode.
Parameters	p0: pointer to the floating point value of the desired relative position of Motor 0 in mm.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadPosition0(float* p0)`	
Description	This function read the current absolute position of the motor0
Parameters	p0: pointer to the floating point value to hold the current absolute position of Motor 0 in mm.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetPosition1(float* p0)`	
Description	This function Set the desired relative position of the motor 1. The motors will run in Position Mode.
Parameters	p0: pointer to the floating point value of the desired relateive position of Motor 1 in mm.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadPosition1(float* p0)`	
Description	This function read the current absolute position of the motor1
Parameters	p0: pointer to the floating point value to hold the current absolute position of Motor 1 in mm.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetAcceleration(float* pA0, float* pA1)`	
Description	This function Set the desired maximum accelerations for motor0 and motor1. The maximum acceleration applies to both velocity mode and position mode.
Parameters	pA0: pointer to the floating point value of the desired maximum acceleration of Motor 0 in mm/sec/sec.

	pA1: pointer to the floating point value of the desired maximum acceleration of Motor 1 in mm/sec/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadAcceleration(float* pA0, float* pA1)`	
Description	This function read the current maximum acceleration of the two motor
Parameters	pA0: pointer to the floating point value to hold the current maximum acceleration of Motor 0 in mm/sec/sec. pA1: pointer to the floating point value to hold the current maximum acceleration of Motor 1 in mm/sec/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetAcceleration0(float* pA0)`	
Description	This function Set the desired maximum accelerations for motor0. The maximum acceleration applies to both velocity mode and position mode.
Parameters	pA0: pointer to the floating point value of the desired maximum acceleration of Motor 0 in mm/sec/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadAcceleration0(float* pA0)`	
Description	This function read the current maximum acceleration for motor0.
Parameters	pA0: pointer to the floating point value to hold the current maximum acceleration of Motor 0 in mm/sec/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetAcceleration1(float* pA0)`	
Description	This function Set the desired maximum accelerations for motor1. The maximum acceleration applies to both velocity mode and position mode.
Parameters	pA0: pointer to the floating point value of the desired maximum acceleration of Motor 0 in mm/sec/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadAcceleration1(float* pA0)`	
Description	This function read the current maximum acceleration for motor1.
Parameters	pA0: pointer to the floating point value to hold the current maximum acceleration of Motor 0 in mm/sec/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetMaxVelocity(float* pV0, float* pV1)`	
Description	This function Set the maximum velocity for motor0 and motor1 for position mode.
Parameters	pV0: pointer to the floating point value of the desired maximum velocity of Motor 0 in mm/sec. pV1: pointer to the floating point value of the desired maximum velocity of Motor 1 in mm/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadMaxVelocity(float* pV0, float* pV1)`	
Description	This function read the current maximum acceleration of the two motor
Parameters	pA0: pointer to the floating point value to hold the current maximum velocity of Motor 0 in mm/sec/sec. pA1: pointer to the floating point value to hold the current maximum velocity of Motor 1 in mm/sec/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetMaxVelocity0(float* pV0)`	
Description	This function Set the maximum velocity for motor0 for position mode.
Parameters	pV0: pointer to the floating point value of the desired maximum velocity of Motor 0 in mm/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadMaxVelocity0(float* pV0)`	
Description	This function read the current maximum velocity of the motor0
Parameters	pV0: pointer to the floating point value to hold the current maximum velocity of Motor 0 in mm/sec/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetMaxVelocity1(float* pV0)`	
Description	This function Set the maximum velocity for motor1 for position mode.
Parameters	pV0: pointer to the floating point value of the desired maximum velocity of Motor 0 in mm/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadMaxVelocity1(float* pV0)`	
Description	This function read the current maximum velocity of the motor1

Parameters	pV0: pointer to the floating point value to hold the current maximum velocity of Motor 0 in mm/sec/sec.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

PWM Mode Functions

`char SetPWM(float* pV0, float* pV1)`	
Description	This function set the PWM duty cycle of the drivers of the two DC motors
Parameters	pV0: pointer to the floating point value of the PWM duty cycle of Motor 0. (-1.0 to 1.0 indicates -100% to 100%) pV1: pointer to the floating point value of the PWM duty cycle of Motor 1. (-1.0 to 1.0 indicates -100% to 100%)
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadPWM(float* pV0, float* pV1)`	
Description	This function set the current PWM duty cycle of the drivers of the two DC motors
Parameters	pV0: pointer to the floating point value to hold the current PWM duty cycle of Motor 0 (-1.0 to 1.0 indicates -100% to 100%). pV1: pointer to the floating point value to hold the current PWM duty cycle of Motor 1 (-1.0 to 1.0 indicates -100% to 100%).
Return Value	1 if the function call is successful 0 if the function call fails

`char SetPWM0(float* pV0)`	
Description	This function set the PWM duty cycle of the driver of the motor 0
Parameters	pV0: pointer to the floating point value of the PWM duty cycle of Motor 0. (-1.0 to 1.0 indicates -100% to 100%)
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadPWM0(float* pV0)`	
Description	This function Read the current PWM duty cycle of the driver of the motor 0
Parameters	pV0: pointer to the floating point value to hold the current PWM duty cycle of Motor 0. (-1.0 to 1.0 indicates -100% to 100%)
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetPWM1(float* pV0)`	
Description	This function set the PWM duty cycle of the driver of the motor 1
Parameters	pV0: pointer to the floating point value of the PWM duty cycle of Motor 1. (-1.0 to 1.0 indicates -100% to 100%)
Return Value	1 if the function call is successful

	0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadPWM1(float* pV0)`	
Description	This function Read the current PWM duty cycle of the driver of the motor 1
Parameters	pV0: pointer to the floating point value to hold the current PWM duty cycle of Motor 1. (-1.0 to 1.0 indicates -100% to 100%)
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

Stop and Reset Functions

`char EStop(void)`	
Description	Stop both motor at the best effort
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char EStop0(void)`	
Description	Stop motor0 at the best effort
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char EStop1(void)`	
Description	Stop motor1 at the best effort
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char MotorReset(void)`	
Description	Reset both motor controller.
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char MotorReset0(void)`	
Description	Reset motor0 controller.
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char MotorReset1(void)`	
Description	Reset motor1 controller.
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char MotorON(void)`	
Description	Turn ON the drivers of both motors. The motor will be in Velocity Mode
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char MotorOFF(void)`	
Description	Turn OFF the drivers of both motors. The motor is in free running mode
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char Motor0ON(void)`	
Description	Turn ON the driver of motor 0. The motor0 will be in Velocity Mode
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char Motor0OFF(void)`	
Description	Turn OFF the driver of motor0. The motor0 is in free running mode
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char Motor1ON(void)`	
Description	Turn ON the driver of motor 1. The motor1 will be in Velocity Mode
Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char Motor1OFF(void)`	
Description	Turn OFF the driver of motor1. The motor1 is in free running mode

Parameters	
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

Wheel Parameter Functions

`char SetWheel0Info(float* Gear, float* Diameter, float* CPR)`	
Description	This function set the wheel configuration for motor 0
Parameters	Gear: pointer to the floating point value of the Gear Ratio. E.g. 13.0 indicates gear reduction ratio of 13:1. Diameter: pointer to the floating point value of the diameter of the wheel in mm. CPR: pointer to the floating point value of the Count Per Revolution of the incremental encoder. E.g. for a 2-channel encoder with CPR = 512, the actual CPR is 512X4 = 2048.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadWheel0Info(float* Gear, float* Diameter, float* CPR)`	
Description	This function read the wheel configuration for motor 0
Parameters	Gear: pointer to hold the floating point value of the Gear Ratio. Diameter: pointer to hold the floating point value of the diameter of the wheel in mm. CPR: pointer to hold the floating point value of the Count Per Revolution of the incremental encoder.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetWheel1Info(float* Gear, float* Diameter, float* CPR)`	
Description	This function set the wheel configuration for motor 1
Parameters	Gear: pointer to the floating point value of the Gear Ratio. E.g. 13.0 indicates gear reduction ratio of 13:1. Diameter: pointer to the floating point value of the diameter of the wheel in mm. CPR: pointer to the floating point value of the Count Per Revolution of the incremental encoder. E.g. for a 2-channel encoder with CPR = 512, the actual CPR is 512X4 = 2048.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadWheel1Info(float* Gear, float* Diameter, float* CPR)`	
Description	This function read the wheel configuration for motor 1
Parameters	Gear: pointer to hold the floating point value of the Gear Ratio. Diameter: pointer to hold the floating point value of the diameter of the wheel in

	mm. CPR: pointer to hold the floating point value of the Count Per Revolution of the incremental encoder.
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

PID Parameter Functions

`char SetPID0Parameters(float* kp, float* ki, float* kd)`	
Description	This function set the PID parameters of the motion controller for motor0
Parameters	kp: pointer to the floating point value of the Proportional Gain ki: pointer to the floating point value of the Integral Gain kd: pointer to the floating point value of the Derivative Gain
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadPID0Parameters(float* kp, float* ki, float* kd)`	
Description	This function read the PID parameters of the motion controller for motor0
Parameters	kp: pointer to hold the floating point value of the Proportional Gain ki: pointer to hold the floating point value of the Integral Gain kd: pointer to hold the floating point value of the Derivative Gain
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char SetPID1Parameters(float* kp, float* ki, float* kd)`	
Description	This function set the PID parameters of the motion controller for motor1
Parameters	kp: pointer to the floating point value of the Proportional Gain ki: pointer to the floating point value of the Integral Gain kd: pointer to the floating point value of the Derivative Gain
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

`char ReadPID1Parameters(float* kp, float* ki, float* kd)`	
Description	This function read the PID parameters of the motion controller for motor1
Parameters	kp: pointer to hold the floating point value of the Proportional Gain ki: pointer to hold the floating point value of the Integral Gain kd: pointer to hold the floating point value of the Derivative Gain
Return Value	1 if the function call is successful 0 if the function call fails
Files	MotionCtrl.h, iMicro1.lib

RC Servo pulse control function

`char SetRCServo(unsigned char ch, int offset)`	
Description	This function set the pulse with of the specified RC pulse channel
Parameters	ch: the channel number (0 – 15) offset: the offset (16-bit signed integer) from the neutral pulse width. Neutral pulse width is 1.5ms. One step of the offset parameter change the pulse width by 0.1uS. For example, offset = 800 will result in a pulse width of 1.58ms.
Return Value	1 if the function call is successful 0 if the function call fails
Files	RCServo.h, iMicro1.lib

Digital Compass function

`char Read_I2C_Register(unsigned char addr, unsigned char len,  unsigned char* buf)`	
Description	This function read the data from the specified registers in the compass CMP03
Parameters	addr: the address of the register len: the length of the data to be read buf: the pointer to the buffer to hold the data
Return Value	1 if the function call is successful 0 if the function call fails
Files	CMP03.h, CMP03.c

`char Read_Degree_16(unsigned int* buf)`	
Description	This function read the 16 bit compass direction
Parameters	buf: the pointer to the buffer to hold the direction data. 2700 means 270.0 degree
Return Value	1 if the function call is successful 0 if the function call fails
Files	CMP03.h, CMP03.c

For details of compass module, please refer to
http://www.robot-electronics.co.uk/htm/cmps3doc.shtml

Analog to Digital Converter function

`void ReadAnalogInput(char ch, far int* data)`	
Description	This function read the 10 bit ADC data
Parameters	Ch: the channel numer (0-7) data: the pointer to the buffer to hold the 10 bit ADC data
Return Value	
Files	ADC.h, iMicro1.lib

Serial Communication Functions

The serial communication library implements the interrupt driven serial input. The implemented buffer size for the serial input is 100 bytes. The benefit of using this buffer is that the serial input data are stored in the buffer while the main program is still busy doing other tasks. This prevents the incoming data loss.
The function to write to serial port is not buffered.

`void ISR_Read_UART(void)`	
Description	This is the interrupt service routine to be placed at the interrupt vector for incoming serial data handling
Parameters	
Return Value	
Files	Serial_com.h, iMicro1.lib

`void Init_Serial_COM(unsigned char BaudRate)`	
Description	This function initialize the serial communication module
Parameters	BaudRate: the baud rate register value to be used to determine the baud rate. The UART of the processor is configured as high-speed (BRGH=1), no-parity, 1-stop bit. Please refer to the PIC18F4620 data sheet for baud rate register configuration
Return Value	
Files	Serial_com.h, iMicro1.lib

`char Read_Serial(far char* pt)`	
Description	This function read one character from the input buffer
Parameters	pt: pointer of character to hold the reading data
Return Value	1: the reading is successful 0: the buffer is empty
Files	Serial_com.h, iMicro1.lib

`char Write_Serial(char c)`	
Description	This function write one character to the UART
Parameters	c: the character to be written
Return Value	1: the transmission is successful 0: the transmission fails
Files	Serial_com.h, iMicro1.lib

`char Write_Serial_S(far char* pt)`	
Description	This function write a NULL terminated string to the UART
Parameters	pt: the pointer of string to be written
Return Value	1: the transmission is successful 0: the transmission fails
Files	Serial_com.h, iMicro1.lib

`char Write_Serial_RS(far const rom char* pt)`

Description	This function write a NULL terminated ROM string to the UART
Parameters	pt: the pointer of string to be written
Return Value	1: the transmission is successful 0: the transmission fails
Files	Serial_com.h, iMicro1.lib

RF Serial Communication Functions

The RF serial communication library implements the transmitting and receiving functions for Radiometrix RF modules. These function call the serial communication functions.
The outgoing data is followed by 10 bytes of preamble (0x55) and 2 byte of header (0xF0).

`void Set_RF_Mode(void)`	
Description	Set the UART multiplexer to RF module. The RS232 is disconnected
Parameters	
Return Value	
Files	RF.h, RF.c

`void Set_RS232_Mode(void)`	
Description	Set the UART multiplexer to RS232. The RF is disconnected
Parameters	
Return Value	
Files	RF.h, RF.c

`unsigned char RF_Read_Data(char* buf, unsigned char len)`	
Description	This function read the data from RF module
Parameters	buf: the pointer to the buffer to hold the incoming data. len: the length of incoming data to be read.
Return Value	The length of incoming data that has been received
Files	RF.h, RF.c

`void RF_Send_Data(char* buf, unsigned char len)`	
Description	This function send the data thru RF module
Parameters	buf: the pointer to the buffer to hold the outgoing data. len: the length of outgoing data to be sent.
Return Value	
Files	RF.h, RF.c