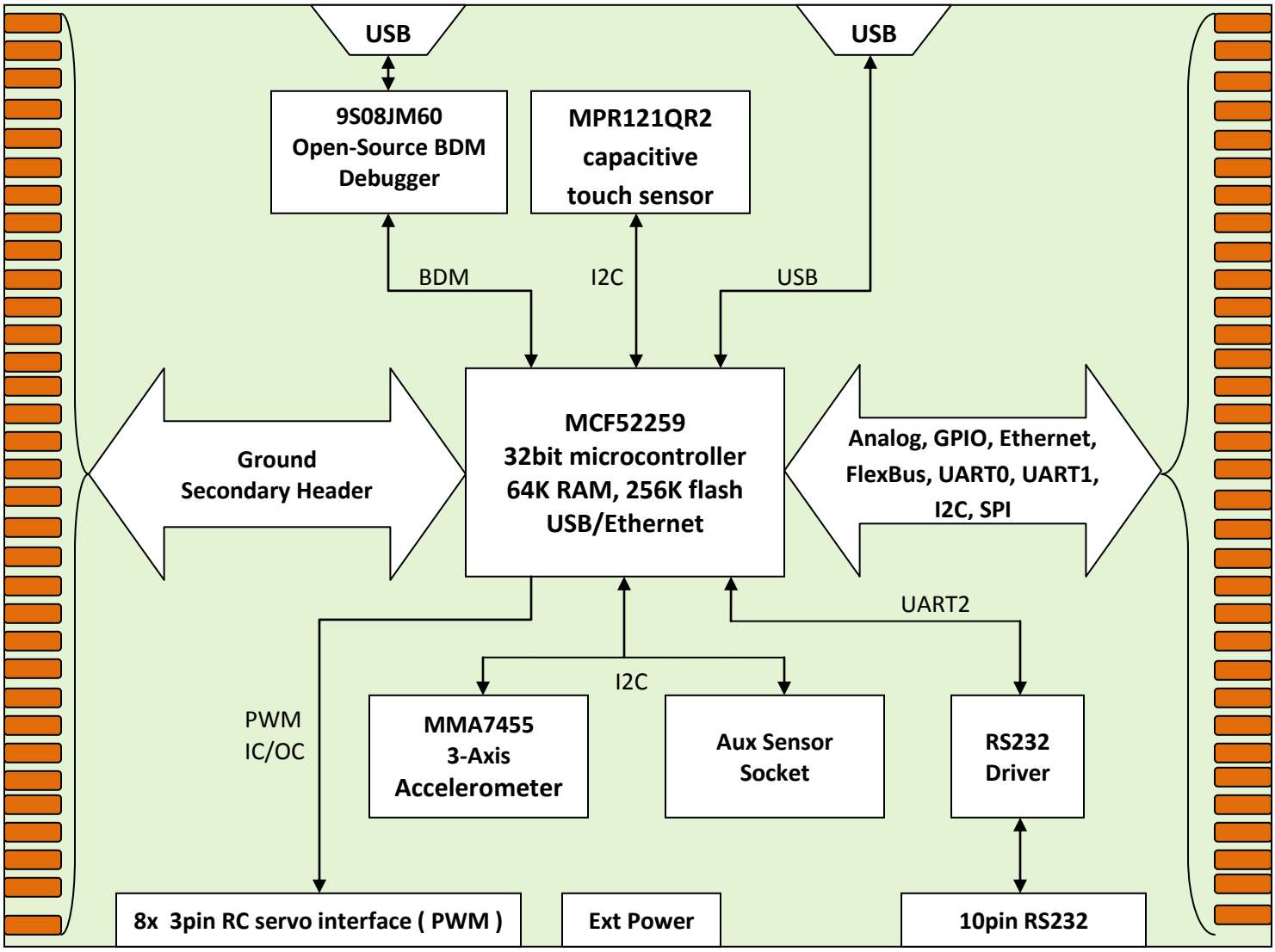
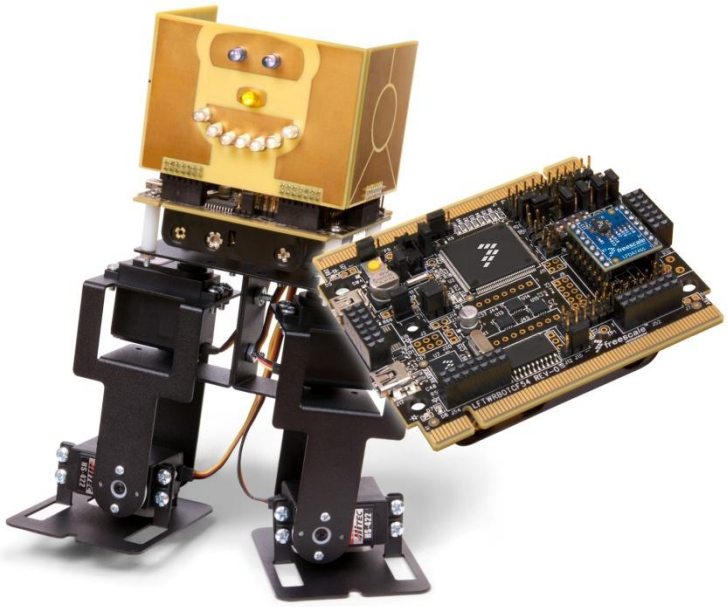
TWR-MECH FSLBOT Codewarrior Sample Code

The Freescale Tower Mechatronic board makes it easy to learn and experiment with sensors and electromechanics. The board has a 32bit Coldfire microcontroller with 64K of RAM, and 512K of flash, and supports the full range of Freescale sensors via plug-in daughter boards. FreeBot is a Sensor development kit in the form of a 4DOF (Degree Of Freedom) bipedal walking robot. Learn how to write software for sensors, while making a robot walk and respond to touch.

Using the on-board OSBDM debugger and free **Codewarrior Special Suite development studio (IDE)**, the board can easily be programmed in either C or assembly (C++ is available with the Codewarrior Professional Suite). Codewarrior Special Suite plus OSBDM is a complete debugging solution including; flash programming, run control (source level single step), and trace.





The **Tower Mechatronics Board / FreeBot Codewarrior Software Toolkit** is a collection of drivers and demo software (all source code included) put together to make it easy to start programming the Mechatronic Board in C. The demo programs are combined in a Codewarrior MCP project file, getting started requires just one mouse click.

Drivers available in the Toolkit:

- General Purpose 16bit Timers (GPT) – Input Capture, Output Compare, and PWM
  - Configurable for .25us resolution fine RC servo control
- PWM controller (8bit)
  - Configurable for 7us resolution course RC servo control
- I2C Master Mode
  - MMA7455 3-axis accelerometer support
  - MPR121 12 channel touch channel support
- QSPI Master Mode
- UART buffered and un-buffered
- Analog to Digital Converter
- Interrupt Controller
- DMA Controller
- FlexCan Controller
- Periodic Interrupt Timers
- DMA Timers

The Demo's in the toolkit (written in C) provide examples of how to use the above drivers to make FreeBot balance on one leg using the 3-axis accelerometer and walk using the 3-axis accelerometer.

Examples of High-level Functions available in the Toolkit:

- **void InitServos(void)**
  Initializes the PWM controller and the GPT controller for RC servo control operation (20ms period with duty cycle between 500us and 2500us).

- **void MMA7455AccelInit(void)**
  Initialize the I2C and MMA7455 3-axis accelerometer.

- **void InitMPR121(void)**
  Initialize the MPR121 for auto scan of all 12 touch channels, using default parameters.

- **void MMA7455AccelRead(uint8 *results)**
  Read the 3 channels of the 3-axis accelerometer and store the values in the byte array pointed to by results.

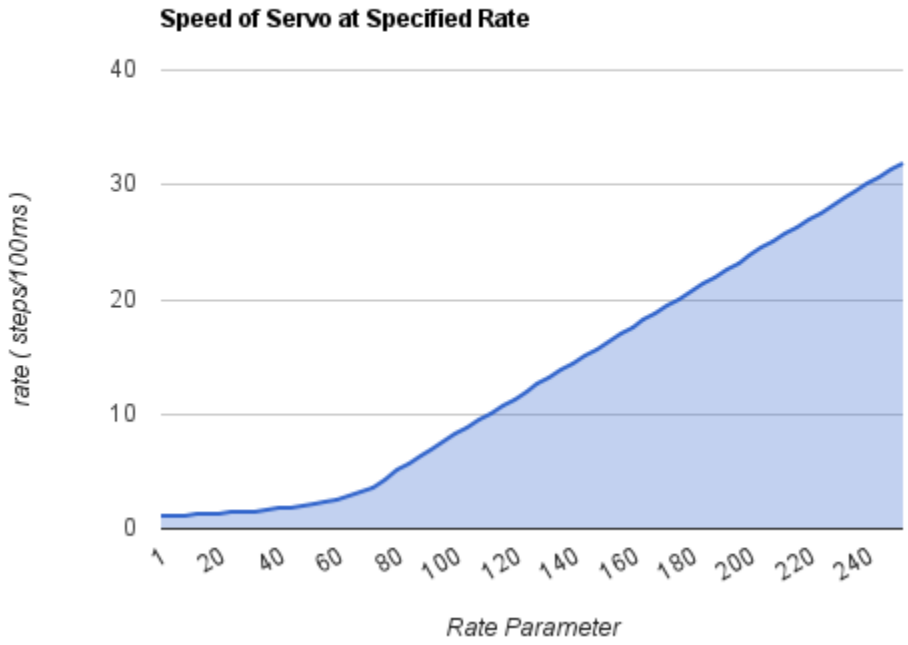
- **uint16 ReadMPR121_Status(void)**
  Read the 16 bit binary channel status of the touch sensor. Returns 12 bits representing the state of the 12 touch channels (1 = touched, 0 = not touched).

- **void MoveServo(uint8 channel, uint8 pos, uint8 rate)**
  Move a single servo (specified by channel) to a position (pos) at a specified rate (rate). Channel 0 is J31, Channel 1 is J33, J35, J37, J30, J32, J34, J36
  Position is a 8bit value representing the position of the servo from 500us to 2500us.
  The function blocks until the servo moves from its current position to the new position.

- **void MoveServos(uint8 servo1, // servo J31 position**
  uint8 servo2, // servo J33 position
  uint8 servo3, // servo J35 position
  uint8 servo4, // servo J37 position
  uint8 servo5, // servo J30 position
  uint8 servo6, // servo J32 position
  uint8 servo7, // servo J34 postion
  uint8 servo8, // servo J36 position
  rate) // rate at which to move servos
  Move all 8 servos at the same time from their current positions to the new desired positions at the desired rate. The function blocks until ALL servos have reached their desired positions. Position is a 8bit value representing the position of the servo from 500us to 2500us.

How the rate parameter effect actual servo speed

The *rate* parameter in the servo commands specifies how fast the servo moves when a new position is specified. The rate parameter is a value from 1 to 255. 1 is the slowest rate and is equal to one 8bit step per 100ms, or 10 steps per second. 255 is the fastest rate, and will move the servo as fast as the servo can move (full speed). Values between 1 and 255 provide servo rates as specified in the following table.



Example Program – Balancing on 1 leg using the 3-axis accelerometer

```
void main (void)
{
        uint8   i, mode, results[3], mode=0;
        uint16  touch;

        (void) PITInit(0, 800, 0);                      // Init PIT for 1ms resolution
        InitServos();                                   // Init PWM and GPT for RC servo control
        printf( "\nTower Mechatronic board\n" );
        MMA7455AccelInit();                             // Initialize the I2C and MMA7455 3-axis accel
        InitMPR121();                                   // Initialize the MPR121 12 channel touch sensor
        while( 1 )                                      // Loop
        {
                MMA7455AccelRead( results );            // Read all 3 channels of the accel sensor
                touch = ReadMPR121_Status();            // Read the 12 channels of the touch sensor
                switch( mode )
                {
                        case 0: // Stand up
                                MoveServos(     CENTER, // J31 servo position
                                                CENTER, // J33 servo position
                                                CENTER, // J35 servo position
                                                CENTER, // J37 servo position
                                                CENTER,CENTER,CENTER,CENTER, // J30,J32,J34,J36 servos
                                                25 );   // rate of motion ( 1 – 255 ) 255 = fastest
                                if( touch & 0x0F00 ) mode = 1; // right ear
                                if( touch & 0x000F ) mode = 2; // left ear
                                break;
                        case 1: // lean left
                                i = (char)results[1]/4;
                                MoveServos(     CENTER, // J31 servo position
                                                CENTER, // J33 servo position
                                                160-i,  // J35 servo position
                                                 CENTER, // J37 servo position
                                                CENTER,CENTER,CENTER,CENTER, // J30,J32,J34,J36 servos
                                                25 ); // rate of motion ( 1 – 255 ) 255 = fastest
                                if( touch & 0x0010 ) mode = 0; // face
                                break;
                        case 2: // lean right
                                i = (char)results[1]/4;
                                MoveServos(     CENTER, // J31 servo position
                                                80-i,   // J33 servo position
                                                CENTER, // J35 servo position
                                                CENTER, // J37 servo position
                                                CENTER,CENTER,CENTER,CENTER, // J30,J32,J34,J36 servos
                                                25 ); // rate of motion ( 1 – 255 ) 255 = fastest
                                if( touch & 0x0010 ) mode = 0; // face
                                break;
                } // switch
        } // while
}
```