The Evolution Robotics ER1 Robot Control Module (RCM) S.N. 100xxx ER1

Unofficial Hardware Description by David Prutchi  www.prutchi.com

The Brains of the ER1 RCM

The Evolution Robotics ER1 RCM is based on the Pilot® MC3410 single-axis, single chip motion processor by Performance Motion Devices Corporation (www.pmdcorp.com). The MC3410 microstepping motion control processor is ideal for embedded systems in industrial control, automation and robotic applications. The chip is a 132-pin device, surface mount CMOS technology and powered by 5 volts. The motion processor is driven by a host microprocessor through an asynchronous bi-directional serial port, giving users the ability to offload resource intensive motion control functions from the application’s host.
The MC3410 outputs the PWM or DAC-compatible motor command signal needed to directly drive the windings of a stepping motor. A programmable microstepping rate can be specified to designate the desired number of microsteps per full step, from 1 to 256. The chip operates in an open loop mode where the motor command is driven from the output of the trajectory generator. Optional encoder feedback provides on-the-fly motor stall detection and allows the chip to detect when the stepping motor has lost steps during a motion. Trace capabilities provide on-the-fly data storage for analyzing system performance and performing maintenance and diagnostics.

With over 115 commands, PMD’s instructions set offers flexibility and versatility to board designers and software application programmers. Instructions are used to initialize and control the motion processor. User-selectable profiling modes supported by the motion processor include S-curve, trapezoidal, velocity contouring and user-defined. The MC3410 accepts input parameters such as position, velocity and acceleration from the host and generates a corresponding trajectory.

The motion processor accepts feedback from an incremental encoder, up to 5 megacounts per second or from an absolute encoder or resolver, up to 160 megacounts per second, to read the current position. 16-bit DAC or 8-bit, 80kHz PWM-compatible output signals are supported.

Multiple breakpoints offer precise sequencing and control of events by the application program. PLC-style instructions are provided, which operate on inputs and set outputs. The instructions use Event, Activity and Signal registers. Input signals include two limit switches (one for each direction of travel), home indicator and a general-purpose programmable input. An at-rest output signal is provided, along with one general-purpose programmable output signal is also provided. Eight general-purpose analog (0-5 V) and 256 (16-bit wide) general-purpose discrete inputs/outputs are available on each Pilot® MC3410.
Motor Drivers

The ER1 motors are driven by Allegro A3959SLB DMOS Full-Bridge PWM Motor Driver ICs. These chips are designed for pulse-width modulated (PWM) current control of dc motors. They are capable of output currents to ±3 A and operating voltages to 50 V. They have low rDS(on) outputs (270 milliohms) for high efficiency. Internal fixed off-time PWM current-control timing circuitry can be adjusted via control inputs to operate in slow, fast, and mixed current-decay modes.

PHASE and ENABLE input terminals are provided for use in controlling the speed and direction of a dc motor with externally applied PWM-control signals. Internal synchronous rectification control circuitry is provided to reduce power dissipation during PWM operation.

Internal circuit protection includes thermal shutdown with hysteresis, undervoltage monitoring of supply and charge pump, and crossover-current protection. Special power-up sequencing is not required.
Power for the ER1 RCM

The ER1 RCM is powered directly by the ER1 battery (12V). Regulated 5V for the RCM circuitry is generated by a Burr Brown (now Texas Instruments) REG104-5 low-noise, low-dropout linear regulator. The chip is rated for 1A output current. The REG104-5 is well protected—internal circuitry provides a current limit which protects the load from damage. Thermal protection circuitry keeps the chip from being damaged by excessive temperature.
USB Interface

The USB interface for the ER1 RCM is based on the FT232 chip by Future Technology Devices International, Ltd. (FTDI). FTDI provides several USB-based interface chips to use when interfacing peripherals to USB-based PCs. The FT232BM is a USB-to-asynchronous serial interface. Conceptually, it behaves like a modem to the peripheral device. The FT232BM provides an asynchronous serial transmit and receive interface as well as modem control signals ready to interface to an asynchronous UART. High baud rates of up to 920K are supported by this chip set and the drivers.

Glue Logic

Most of the glue logic in the ER1 RCM is based around simple 74LS-family chips. The 74LS244 tri-state octal buffer is prominently used in the interface of the Pilot® MC3410 ICs.
Direct Control of the ER1 RCM

Yau Lam Yiu reverse-engineered the control code for the ER1 RCM. His code is available at [http://www.cs.ust.hk/~yiu/robot/](http://www.cs.ust.hk/~yiu/robot/). A log of some simple motion commands shows the way in which a program would directly communicate with the ER1 RCM without going through Evolution Robotics’ software:

1st column is serial port communication counter,
2nd column is the serial port commands:
   i. IOCTL_SERIAL_* - serial port control command
   ii. IRP_MJ_WRITE  - send data to the serial port
   iii. IRP_MJ_READ  - read data from the serial port
3rd column actual data sending to the serial port.
   i. IRP_MJ_WRITE  - Pilot Motion Processor command
      1st byte: address
      2nd byte: checksum
      3nd byte: axis (always 00)
      4th byte: name of command (command code)
      5th byte+: parameters of the command. (higher order bytes send first)
   ii. IRP_MJ_READ   - Bytes return from the Motion Processor

The log captures the following communication:
1. Start RCMotion
2. press the forward key, i.e. 8,
3. press the stop key, i.e. 5,
4. quit RCMotion

```
0  IRP_MJ_CREATE Options: Open
1  IOCTL_SERIAL_SET_BAUD_RATE
2  IOCTL_SERIAL_GET_LINE_CONTROL
3  IOCTL_SERIAL_GET_CHARS
4  IOCTL_SERIAL_GET_HANDFLOW
5  IOCTL_SERIAL_SET_BAUD_RATE Rate: 250000
6  IOCTL_SERIAL_CLR_RTS
7  IOCTL_SERIAL_CLR_DTR
8  IOCTL_SERIAL_SET_LINE_CONTROL StopBits: 1 Parity: NONE WordLength:
   8
9  IOCTL_SERIAL_SET_CHAR EOF:0 ERR:0 BRK:0 EVT:0 XON:0 XOFF:0
10 IOCTL_SERIAL_SET_HANDFLOW Shake:0 Replace:0 XonLimit:0 XoffLimit:0
11 IOCTL_SERIAL_SET_TIMEOUTS RI:0 RM:1 RC:1 WM:0 WC:0
12 IRP_MJ_WRITE  01 C6 00 Reset(39)
13 IRP_MJ_READ   Length 2: 01 FF
```
14 IRP_MJ_WRITE 01 7F 00 SetLimitSwitchMode(80) 00 00
15 IRP_MJ_READ Length 2: 00 00
16 IRP_MJ_WRITE 01 88 00 SetMotorCommand(77) 00 00
17 IRP_MJ_READ Length 2: 00 00
18 IRP_MJ_WRITE 01 5E 00 SetProfileMode(A0) 00 01
19 IRP_MJ_READ Length 2: 00 00
20 IRP_MJ_WRITE 01 BF 00 SetAcceleration(90) 00 00 00 B0
21 IRP_MJ_READ Length 2: 00 00
22 IRP_MJ_WRITE 01 BE 00 SetDeceleration(91) 00 00 00 B0
23 IRP_MJ_READ Length 2: 00 00
24 IRP_MJ_WRITE 01 FB 00 SetJerk(13) 03 29 0A BB
25 IRP_MJ_READ Length 2: 00 00
26 IRP_MJ_WRITE 00 C7 00 Reset(39)
27 IRP_MJ_READ Length 2: 01 FF
28 IRP_MJ_WRITE 00 80 00 SetLimitSwitchMode(80) 00 00
29 IRP_MJ_READ Length 2: 00 00
30 IRP_MJ_WRITE 00 89 00 SetMotorCommand(77) 00 00
31 IRP_MJ_READ Length 2: 00 00
32 IRP_MJ_WRITE 00 5F 00 SetProfileMode(A0) 00 01
33 IRP_MJ_READ Length 2: 00 00
34 IRP_MJ_WRITE 00 C0 00 SetAcceleration(90) 00 00 00 B0
35 IRP_MJ_READ Length 2: 00 00
36 IRP_MJ_WRITE 00 BF 00 SetDeceleration(91) 00 00 00 B0
37 IRP_MJ_READ Length 2: 00 00
38 IRP_MJ_WRITE 00 FC 00 SetJerk(13) 03 29 0A BB
39 IRP_MJ_READ Length 2: 00 00
40 IRP_MJ_WRITE 01 E5 00 Update(1A)
41 IRP_MJ_READ Length 2: 00 00
42 IRP_MJ_WRITE 00 E6 00 Update(1A)
43 IRP_MJ_READ Length 2: 00 00
44 IRP_MJ_WRITE 01 7C 00 SetMotorCommand(77) 4C C0
45 IRP_MJ_READ Length 2: 00 00
46 IRP_MJ_WRITE 00 7D 00 SetMotorCommand(77) 4C C0
47 IRP_MJ_READ Length 2: 00 00
48 IRP_MJ_WRITE 01 D3 00 SetVelocity(11) FF FC 20 00
49 IRP_MJ_READ Length 2: 00 00
50 IRP_MJ_WRITE 00 0C 00 SetVelocity(11) 00 03 E0 00
51 IRP_MJ_READ Length 2: 00 00
52 IRP_MJ_WRITE 01 E5 00 Update(1A)
53 IRP_MJ_READ Length 2: 00 00
54 IRP_MJ_WRITE 00 E6 00 Update(1A)
55 IRP_MJ_READ Length 2: 00 00
56 IRP_MJ_WRITE 01 7C 00 SetMotorCommand(77) 4C C0
57 IRP_MJ_READ Length 2: 00 00
58 IRP_MJ_WRITE 00 7D 00 SetMotorCommand(77) 4C C0
59 IRP_MJ_READ Length 2: 00 00
<table>
<thead>
<tr>
<th></th>
<th>IRP_MJ_WRITE</th>
<th>01 EE 00 SetVelocity(11) 00 00 00 00</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>IRP_MJ_READ</td>
<td>Length 2: 00 00</td>
</tr>
<tr>
<td>61</td>
<td>IRP_MJ_WRITE</td>
<td>00 EF 00 SetVelocity(11) 00 00 00 00</td>
</tr>
<tr>
<td>62</td>
<td>IRP_MJ_READ</td>
<td>Length 2: 00 00</td>
</tr>
<tr>
<td>63</td>
<td>IRP_MJ_WRITE</td>
<td>01 E5 00 Update(1A)</td>
</tr>
<tr>
<td>64</td>
<td>IRP_MJ_READ</td>
<td>Length 2: 00 00</td>
</tr>
<tr>
<td>65</td>
<td>IRP_MJ_WRITE</td>
<td>00 E6 00 Update(1A)</td>
</tr>
<tr>
<td>66</td>
<td>IRP_MJ_READ</td>
<td>Length 2: 00 00</td>
</tr>
<tr>
<td>67</td>
<td>IRP_MJ_READ</td>
<td>Length 2: 00 00</td>
</tr>
</tbody>
</table>