Switcher

Ashley J. Robinson

January 3, 2017

1 Introduction

2 Hardware

3 Software

3.1 Peripherals

3.1.1 Analogue to Digital Convertors (ADCs)

A single ADC is multiplexed to measure three voltages on the board.

3.1.2 Universal Asyncronhouse/Syncronhous Transceiver (UART)

The UART is configure to run at 115200 baud with no control flow with a ring buffer interface. Trasmission from the microcontroller happens through **uartLoadOut** which adds to the **8 byte** buffer and is then unloaded from a timer. The input is not interrupt driven and is handled in the main control loop using a **5 byte** buffer that is enough to contain the longest command.

3.1.3 Programmable Counter Array (PCA)

The Pulse Width Modulation (PWM) is controlled from the counter array. The output runs at approximatly 96KHz with 8 bits to control the duty cycle. A high resolution for control would be favourable for this application but the frequency achieved in 16-bit mode is far too low for this application.

3.1.4 Timers

Timer 0 is used as baud rate generagtion for the UART. **Timer 2** is used to trigger an Interrupt Service Routine (ISR) which runs at 4KHz. The ISR control the UART transmission, sampling of the ADC, running the controller and finally setting the PWM.

3.2 Operation

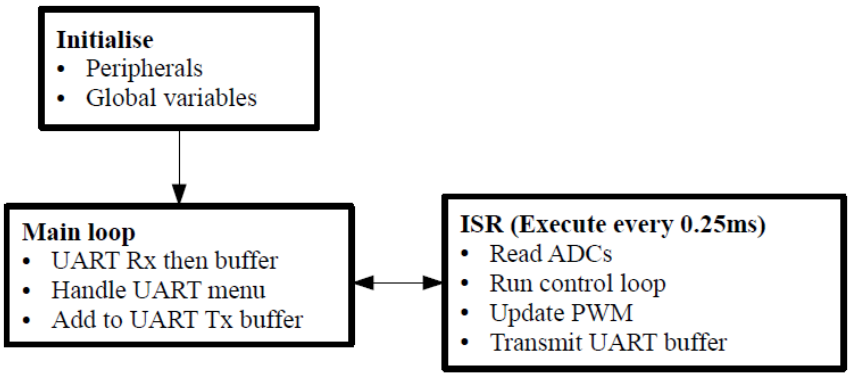

Figure 1: Software overview.

3.3 UART Menu

Command	Char	Send numbers	Return numbers	Notes
Enable	g	0	0	0
Disable	s	0	0	0
Read ADC1	x	0	4	Result in mV
Read ADC2	y	0	4	Result in mV
Read ADC3	z	0	4	Result in mV
Read output current	j	0	4	Result in mA
Set output voltage	v	4	0	Send value in mV
Set otuput current	c	4	0	Send value in mA
Set controller P	p	4	0	Value is divide by 10
Set controller I	i	4	0	Value is divide by 10
Set input voltage upper limit	u	4	0	Send value in mV
Set input voltage lower limit	l	4	0	Send value in mV

Table 1: UART menu

3.4 Assembly analysis

The following sections of code are library function inserted by the compilier implict to facilite some more complex operations.

3.4.1 ADC scaling

Listing 1 is a line of code operating on a 32 bit signed integer and invokes listing 2 and 3. The purpose of this operation is to cast the voltage recorded by the ADC to a representation in mV. It is possible for the target voltage to be translated instead but this would still require the same piece of code. The 12 bit oputput but the ADC must be multiplied 5.926 as to

represent the voltage at the top of the potential divider. Fixed but multiplication followed by a shift operation removes the need for a floating point operation but yeilds the same result.

Listing 1: LIMUL

```
return (((U32)ADC0)*SCALE_MUL) >> 10;
```

Listing 2: LIMUL

```
                C?LIMUL:
C:0x0ACC        EC          MOV         A,R4
C:0x0ACD        8EF0        MOV         0xF0,R6
C:0x0ACF        A4          MUL         AB
C:0x0AD0        CC          XCH         A,R4
C:0x0AD1        C5F0        XCH         A,0xF0
C:0x0AD3        CC          XCH         A,R4
C:0x0AD4        CD          XCH         A,R5
C:0x0AD5        F8          MOV         R0,A
C:0x0AD6        EF          MOV         A,R7
C:0x0AD7        A4          MUL         AB
C:0x0AD8        CE          XCH         A,R6
C:0x0AD9        C5F0        XCH         A,0xF0
C:0x0ADB        2D          ADD         A,R5
C:0x0ADC        FD          MOV         R5,A
C:0x0ADD        E4          CLR         A
C:0x0ADE        3C          ADDC        A,R4
C:0x0ADF        FC          MOV         R4,A
C:0x0AE0        E8          MOV         A,R0
C:0x0AE1        A4          MUL         AB
C:0x0AE2        2E          ADD         A,R6
C:0x0AE3        C8          XCH         A,R0
C:0x0AE4        C5F0        XCH         A,0xF0
C:0x0AE6        3D          ADDC        A,R5
C:0x0AE7        FD          MOV         R5,A
C:0x0AE8        E4          CLR         A
C:0x0AE9        3C          ADDC        A,R4
C:0x0AEA        FC          MOV         R4,A
C:0x0AEB        EF          MOV         A,R7
C:0x0AEC        A4          MUL         AB
C:0x0AED        FF          MOV         R7,A
C:0x0AEE        E5F0        MOV         A,0xF0
C:0x0AF0        28          ADD         A,R0
C:0x0AF1        FE          MOV         R6,A
C:0x0AF2        E4          CLR         A
C:0x0AF3        3D          ADDC        A,R5
C:0x0AF4        FD          MOV         R5,A
C:0x0AF5        E4          CLR         A
C:0x0AF6        3C          ADDC        A,R4
C:0x0AF7        FC          MOV         R4,A
C:0x0AF8        22          RET
```

Listing 3: ULSHR

```
                C?ULSHR:
C:0x0A93        E8          MOV         A,R0
C:0x0A94        600F        JZ          C:0AA5
C:0x0A96        EC          MOV         A,R4
C:0x0A97        C3          CLR         C
C:0x0A98        13          RRC         A
```

```
C:0x0A99    FC      MOV     R4,A
C:0x0A9A    ED      MOV     A,R5
C:0x0A9B    13      RRC     A
C:0x0A9C    FD      MOV     R5,A
C:0x0A9D    EE      MOV     A,R6
C:0x0A9E    13      RRC     A
C:0x0A9F    FE      MOV     R6,A
C:0x0AA0    EF      MOV     A,R7
C:0x0AA1    13      RRC     A
C:0x0AA2    FF      MOV     R7,A
C:0x0AA3    D8F1    DJNZ    R0,C:0A96
C:0x0AA5    22      RET
```

3.4.2 ADC scaling

Listing 4: IMUL

```
            C?IMUL:
C:0x0A2C    EF      MOV     A,R7
C:0x0A2D    8DF0    MOV     0xF0,R5
C:0x0A2F    A4      MUL     AB
C:0x0A30    A8F0    MOV     R0,0xF0
C:0x0A32    CF      XCH     A,R7
C:0x0A33    8CF0    MOV     0xF0,R4
C:0x0A35    A4      MUL     AB
C:0x0A36    28      ADD     A,R0
C:0x0A37    CE      XCH     A,R6
C:0x0A38    8DF0    MOV     0xF0,R5
C:0x0A3A    A4      MUL     AB
C:0x0A3B    2E      ADD     A,R6
C:0x0A3C    FE      MOV     R6,A
C:0x0A3D    22      RET
```

3.4.3 Division

Division and modulus operations are contained in 5 invoke the functions contained in 6.

Listing 5: Division

```
scale /= 10;

num = out / scale;

out %= scale;
```

Listing 6: UDIV

```
            C?UIDIV:
C:0x0A3E    BC000B  CJNE    R4,#0x00,C:0A4C
C:0x0A41    BE0029  CJNE    R6,#0x00,C:0A6D
C:0x0A44    EF      MOV     A,R7
C:0x0A45    8DF0    MOV     0xF0,R5
C:0x0A47    84      DIV     AB
C:0x0A48    FF      MOV     R7,A
C:0x0A49    ADF0    MOV     R5,0xF0
C:0x0A4B    22      RET
C:0x0A4C    E4      CLR     A
```

```
C:0x0A4D    CC        XCH     A,R4
C:0x0A4E    F8        MOV     R0,A
C:0x0A4F    75F008    MOV     0xF0,#0x08
C:0x0A52    EF        MOV     A,R7
C:0x0A53    2F        ADD     A,R7
C:0x0A54    FF        MOV     R7,A
C:0x0A55    EE        MOV     A,R6
C:0x0A56    33        RLC     A
C:0x0A57    FE        MOV     R6,A
C:0x0A58    EC        MOV     A,R4
C:0x0A59    33        RLC     A
C:0x0A5A    FC        MOV     R4,A
C:0x0A5B    EE        MOV     A,R6
C:0x0A5C    9D        SUBB    A,R5
C:0x0A5D    EC        MOV     A,R4
C:0x0A5E    98        SUBB    A,R0
C:0x0A5F    4005      JC      C:0A66
C:0x0A61    FC        MOV     R4,A
C:0x0A62    EE        MOV     A,R6
C:0x0A63    9D        SUBB    A,R5
C:0x0A64    FE        MOV     R6,A
C:0x0A65    0F        INC     R7
C:0x0A66    D5F0E9    DJNZ    0xF0,C:0A52
C:0x0A69    E4        CLR     A
C:0x0A6A    CE        XCH     A,R6
C:0x0A6B    FD        MOV     R5,A
C:0x0A6C    22        RET
C:0x0A6D    ED        MOV     A,R5
C:0x0A6E    F8        MOV     R0,A
C:0x0A6F    F5F0      MOV     0xF0,A
C:0x0A71    EE        MOV     A,R6
C:0x0A72    84        DIV     AB
C:0x0A73    20D21C    JB      0xD0.2,C:0A92
C:0x0A76    FE        MOV     R6,A
C:0x0A77    ADF0      MOV     R5,0xF0
C:0x0A79    75F008    MOV     0xF0,#0x08
C:0x0A7C    EF        MOV     A,R7
C:0x0A7D    2F        ADD     A,R7
C:0x0A7E    FF        MOV     R7,A
C:0x0A7F    ED        MOV     A,R5
C:0x0A80    33        RLC     A
C:0x0A81    FD        MOV     R5,A
C:0x0A82    4007      JC      C:0A8B
C:0x0A84    98        SUBB    A,R0
C:0x0A85    5006      JNC     C:0A8D
C:0x0A87    D5F0F2    DJNZ    0xF0,C:0A7C
C:0x0A8A    22        RET
C:0x0A8B    C3        CLR     C
C:0x0A8C    98        SUBB    A,R0
C:0x0A8D    FD        MOV     R5,A
C:0x0A8E    0F        INC     R7
C:0x0A8F    D5F0EA    DJNZ    0xF0,C:0A7C
C:0x0A92    22        RET
```

3.4.4 Switch

This library function is generated to control large switch statements??

Listing 7: CCASE

```
  C?CCASE:
C:0x0AA6  D083  POP   0x83
C:0x0AA8  D082  POP   0x82
C:0x0AAA  F8    MOV   R0,A
C:0x0AAB  E4    CLR   A
C:0x0AAC  93    MOVC  A,@A+DPTR
C:0x0AAD  7012  JNZ   C:0AC1
C:0x0AAF  7401  MOV   A,#0x01
C:0x0AB1  93    MOVC  A,@A+DPTR
C:0x0AB2  700D  JNZ   C:0AC1
C:0x0AB4  A3    INC   DPTR
C:0x0AB5  A3    INC   DPTR
C:0x0AB6  93    MOVC  A,@A+DPTR
C:0x0AB7  F8    MOV   R0,A
C:0x0AB8  7401  MOV   A,#0x01
C:0x0ABA  93    MOVC  A,@A+DPTR
C:0x0ABB  F582  MOV   0x82,A
C:0x0ABD  8883  MOV   0x83,R0
C:0x0ABF  E4    CLR   A
C:0x0AC0  73    JMP   @A+DPTR
C:0x0AC1  7402  MOV   A,#0x02
C:0x0AC3  93    MOVC  A,@A+DPTR
C:0x0AC4  68    XRL   A,R0
C:0x0AC5  60EF  JZ    C:0AB6
C:0x0AC7  A3    INC   DPTR
C:0x0AC8  A3    INC   DPTR
C:0x0AC9  A3    INC   DPTR
C:0x0ACA  80DF  SJMP  C:0AAB
```