Automatique Electronique ESPE

Projet Interdisciplinaire

Buck Low Level Architecture

Authors :
Xavier Bourlot
Thibault Jean
Clarisse Calmo
Xiaohu Zhang
5AE-ESPE

Supervisors :
T. Rocacher
A. Boyer
G. Auriol
C. Escriba
J. Shea

November 6, 2020

Contents

Chapter 1

Introduction

In the Smart Power Management System, the Buck plays a central role in order to convert the energy. Our previous report explained the high level architecture of the Buck with a reminder of specifications, a list of useful signals and the main blocks on a diagram. Now, we are going to detail the buck schematic, components sizing and the software organisation.

In the hardware part, all the components are divided into two part: the power components and control functions. For the power components, we designed a preliminary switch circuit and determined the switching frequency to 150 kHz. Basing on this condition, considering other constraints, we can confirm the dimensions of components. We also design several functions which can realise the regulation to the circuit. With all the dimensions we got, we can finally build the circuit model and analog with software to inspect if the dimensions are appropriate. In the end, we build our complete schematic with Altium.

With the software part, we allow the customer to change the mode of this system according to their requirements. The customers have four options available: CV, CC, MPPT and Open loop mode. We realise the internal part and the external part by configuring the variables and functions in the software part.

Chapter 2

Hardware

In this section, we will explain our approach and design choices for components sizing, and give our preliminary schematic, edited in Altium.

1. General system overview

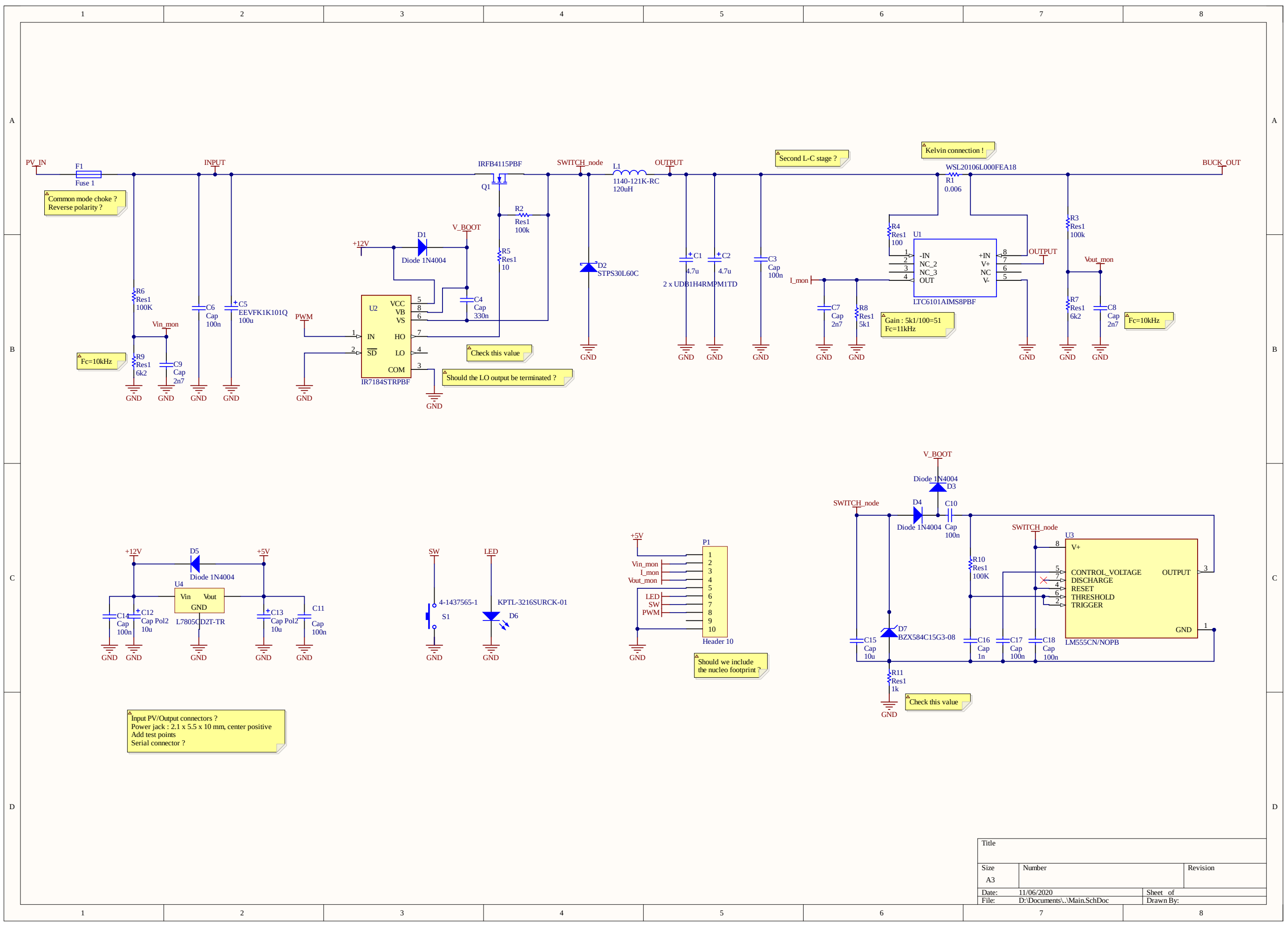


Figure 2.1: Low Level Hardware Architecture

- Top half, from left to right : power flow from the solar panel input to the buck output (BMS input).
- Bottom left : Auxiliary 5V power supply derived from 12V
- Bottom center : microcontroller connector & associated components
- Bottom right : charge pump for the bootstrap supply

2. Component selection & sizing

The following section details some of the component choices and calculations for the critical points of the design. It derives the main requirements and highlights some of the losses that the design will inherently present. No overall efficiency figure is proposed in this document, because not every loss of the design has been estimated yet, rendering any estimation difficult. However, with the main static losses taken into account, the efficiency at 150W would be less than 96% (7.25W loss).

2.1. Power components

The switching frequency has been selected to 150kHz, a trade-off between large components at low frequencies and higher switching losses at high frequencies. This frequency will be used for the calculations in the next paragraphs.

2.1.a. Inductor

The inductor is designed for 700mA worst case ripple current. For the worst case scenario (Vin=48V, Vout=24V), the minimum value is $120\mu H$. Typical current ripple will be less than 500mA. The 1140-121K-RC will be selected as it satisfies the requirements for value and RMS current, and is already available in stock. It's DCR is $28m\Omega$, which results in static loss of 2.27W.

2.1.b. Cout

The output capacitor must satisfy the following constraints :

- Rated voltage $> 40V$
- $I_{RMS} > 700mA$
- $C > 3\mu F$ for a maximum output voltage ripple of 0.5V
- C small enough to allow connection to the batteries & minimize overshoot at no load or during transients

Given these requirements, it is difficult to find a small value capacitor with sufficiently high I_{RMS}.

The UDB1H4RMPM1TD is a $4.7\mu F$ aluminium electrolytic capacitor, $50V$, with $tan\delta = 0.15$ and $I_{RMS} = 400mA$. We can estimate the ESR given the $tan\delta$: $ESR = 1\Omega$. Two identical capacitors in parallel will double the capacitance and halve the ESR, and meet the

I_{RMS} requirements. Using the same capacitors in parallel (and possibly matching their value) will guarantee better sharing of the ripple current between the two devices.

The expected output voltage ripple with this configuration, accounting for the ESR contribution, is 360mV, which meets our 500mV target.

The expected response of the L-C filter is shown in appendix, as well as the LTSpice schematic used.

2.1.c. MOSFET

The mosfet must have a low on-resistance, hold the maximum PV voltage and max current. The IRFB4115PBF fits these needs and is already in stock. At the worst case duty cycle (67%), the static power loss is approx 0.4W, due to the $Rds_{ON} = 11m\Omega$. The estimated temperature of the case without any heatsink is then 50°C, but this does not take dynamic losses into account, which could be the main losses at stake here. These will be calculated later on to determine if heat-sinking the device is necessary.

2.1.d. Diode

The diode must have a low forward voltage to minimize power dissipation, be reasonably fast, allow for the peak current of 10A and hold a reverse voltage of 50V max. The average current through the diode is 6.75A for the worst case scenario (Duty cycle of 25% and full output current). With a typical forward voltage of 450mV at the given current, the power dissipation is approx. 3W.

Depending on the package of the chosen diode, a small heatsink may be required. By example, the STPS30L60C surface mount dual diode may satisfy the electrical requirements, but a $5cm^2$ min area of copper would be required to maintain the junction temperature of the diode at $125 \deg C$, given a thermal resistance of $35 \deg C/W$. A TO-220 package may be better suited, as it can be mounted on a heatsink.

2.2. Other components

The input capacitor, protections (such as reverse polarity, EMI suppression choke, overload) are still in design and therefore not properly sized or present on the schematic for the moment. The output secondary filter inductor is still the design phase as well.

2.3. Control functions

Aside from the power path, several functions are required to ensure proper regulation. The calculations and choices are detailed in the following paragraph.

2.3.a. Vin & Vout sense

Two voltage dividers are used to sense the input and output voltage. Output voltage monitoring allows for constant voltage regulation, and input monitoring allows for MPPT regulation, combined with current sensing.

To ensure protection of the microcontroller analog inputs, the desired maximum measurable voltage is set to 55V, and the resistor divider current to $500\mu A$ (an order of magnitude higher than the ADC input pin current). A capacitor is placed at the ADC input to filter the switching frequency ($F_{-3dB} = 10kHz$). The following standard values are obtained :

$$\begin{cases} R_1 = 6.2k\Omega \\ R_2 = 100k\Omega \\ C = 2.7nF \end{cases}$$

The same circuit will be used for both input and output voltage sensing, to minimize the number of different component values. The ADC's 12bit resolution allows sufficient granularity in both measurements for proper regulation. Due to the small current through the divider, very little power is lost by the measurement.

$$\begin{cases} Vin_{max} = 56.5V \\ Resolution = 14mV \\ P_{loss} = 2 * 30 = 60mW \end{cases}$$

2.3.b. Current sense

To sense the current, a shunt resistor will be used, with a corresponding current sense amplifier. The high side topology as been selected, despite of its higher common mode requirement, to ensure a single common ground across all the system, which eases probing and integration with the BMS.

The shunt resistor is selected to minimize power dissipation whilst maximizing dynamic range. WSL20106L000FEA18 is selected, as it is already available in stock and presents a good trade-off. This low ohmic value requires a careful layout, making use of Kelvin connections to ensure good accuracy.

$$\begin{cases} R_{shunt} = 6m\Omega \\ P_{shunt} = 600mW \\ G_{max} = 55 \end{cases}$$

The maximum gain is calculated to not saturate the ADC input at the maximum current of 10A. The selected current sense amplifier is the LTC6101. Here is why it fits the requirements of the application :

- Common mode requirements : measure 60mV over 24V : common mode of $52dB$.

  LTC6101 has a CMRR of 115dB min, 140dB typ.

  So this gives an output ENOB = $\frac{115-52}{6.02}$ = 10.4bits min, 14.6bits typ.

  The ADC is 12bits, which yields a current resolution of $\frac{10A}{2^{12}} = 2.4mA$, or 9.76mA for the effective 10bits $\rightarrow$10 bits is enough resolution.

- Input offset voltage of $150\mu V$max, so $\frac{150\mu V}{60mV} * 10A = 25mA$ equivalent to the measurement (typ. offset : $85\mu V$, 14mA equ.)

- Gain settable with 2 resistors

- Vin_{max} >PV max output voltage : can't be blown up even in the worst case scenario

- Current consumption $< 1mA$

- \$2 in single quantity, readily available from common distributors

The chosen amplifier gain will be 51, and the output current capability of the amplifier will be set to 1mA. Referring to the amplifiers datasheet, this gives :

$$\begin{cases} V_{ADCFS} = 3.06V \\ R_1 = 100\Omega \\ R_2 = 5.1k\Omega \end{cases}$$

2.3.c. MOS driver

The MOS driver is imposed by the specification. A pull down resistor is added to the gate of the mosfet to ensure it is turned off by default. A series resistor is added to the gate to trim the turn on/off time of the MOS and limit the driver output current.

The N-MOS has to be driven with a floating voltage because its source is on the switching node. Therefore, a bootstrap supply has to be generated from the auxiliary 12V supply. The bootstrap capacitor is selected given the gate minimum voltage of 7V and charge (120nC) of the mosfet.

$$C > \frac{2(2 * Q_g + \frac{I_{qbs}}{F_{sw}})}{V_{cc} - Vf - Vmin} = \frac{2 * (2 * 120nC + \frac{150\mu A)}{100kHz}}{12 - 0.7 - 7} = 112nF$$

A 330nF capacitor will be selected. The bootstrap diode must be able to withstand approx 50V and $120nC * 100kHz = 120mA$, with little reverse leakage. The 1N4148 with its 200mA limit and fast response time is enough for this application.

2.3.d. Bootstrap supply

An external bootstrap supply will be provided to ensure proper MOS drive even at 100% duty cycle. The implementation will reuse an existing design based around a floating 555 IC.

2.3.e. Power supply

From the auxiliary 12V supply, a 5V supply is derived to power the microcontroller. A linear regulator is used, with its associated decoupling capacitors. No reverse input polarity protection is implemented because the power jack can't physically be reversed.

Chapter 3

Software organisation

In this section, we will give a detailed architecture of the software part.

1. Architecture overview

First, here is a class diagram of the software organisation Figure 3.1. In this diagram we can see two main parts referring to the internal processes and the external API. We will comment and explain this diagram in the next sections.

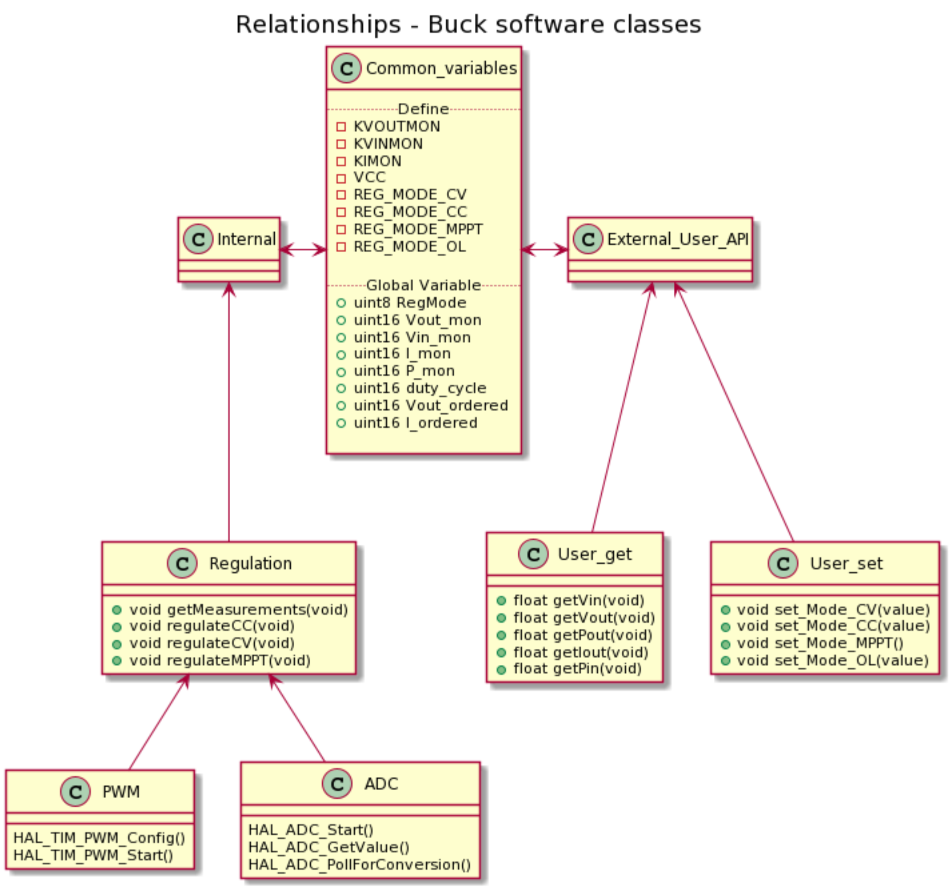


Figure 3.1: Low Level Software Architecture

2. Software layers

If we analyse the diagram by rows, we can notice that each rows refers to a software layer.

2.1. High level layer

The first row at the top refers to the high level layer, with on one hand the user interface, and on the other hand the internal management. These two parts do not communicate with a direct relationship, but through shared variables.

2.2. Middle level layer

The second row in the middle refers to the different functions used to meet the Buck specifications. On the left side, we can see the internals fonctions used for the regulation (CC, CV or MPPT). On the right side, we have the functions available in the user API, for getting signal values or for setting a specific mode.

2.3. Low level layer

The third and last row refers to the hardware layer. Theses classes are located on the left because this layer only makes sense in the internal side that communicates with the physical board. This last layer is about communication with the buck, so we will use HAL functions for the PWM or the ADC.

3. Contants and variables

In the first row of the Figure 3.1, the central block contains our constants and variables.

3.1. Constants

We set several "Define" for constants values used all along the code. The three first values refers to signals values coefficient conversion. *KVOUTMON* and *KVINMON* refer to the divider bridges at the input and the output of the buck, in order to fit the microcontroller voltage range. Next, we have *KIMON* that helps reading the output current thanks to the shunt resistance, and *VCC* which is 3.3V. Finally, the last four constants are the regulation mode values, *REG_MODE_CV, REG_MODE_CC, REG_MODE_MPPT, REG_MODE_OL* (OL for Open Loop), which are respectively 0,1,2,3.

3.2. Variables

In addition, we have several variables that can be modified by the internal and the external part as mentioned before. With these variables, we are able to store and update the values of the current regulation mode with *RegMode*, or the different signals values *Vout_mon, Vin_mon, I_mon.* We also compute and store the output power *P_mon* and the duty cyle *duty_cyle.* Finally, we have the targeted voltage *Vout_ordered* and current I_ordered values used for CV and CC regulation.

4. Classes and functions

4.1. Regulation Class

In this class, we use four functions aimed to regulate the buck. First, the regulation implies having the different signals values so the function `getMeasurements()` recovers thoses values. Secondly, we have three regulation functions, `regulateCC(), regulateCV(), regulateMPPT()` that adjust the duty cycle value. Regarding the code architecture, we will implement a switch case block that reads the value of the mode variable *RegMode*, and then calls the associated regulation function. These functions are type void and they do not take any parameter.

4.2. User Classes

User classes are separated in two parts : one for accessing signal values, and the other for setting a regulation mode.

4.3. Getter

We have five functions available in the API, so that the user can recover the value of :
- the output current `getIout()`
- the input and output voltage `getVin()` and `getVout()`
- the input and output power `getPin()` and `getPout()`

These functions return a float and they do not take any parameter. They work by reading the the actual signal value stored in the global variables. Therefore, they do not directly read the ADC but return the last sampled value.

4.4. Setter

We have four function available in the API, so that the user can choose which regulation mode to apply :
- CV regulation `set_Mode_CV(value)`
- CC regulation `set_Mode_CC(value)`
- MPPT regulation `set_Mode_MPPT()`
- Open loop mode `set_Mode_OL(value)`

These functions are of type void because they do not return anything, these functions are there to allow the client/users to change the regulation mode of the system so we have chosen not to return anything because the client will see the change live on the physical system. Each of these functions take as argument a variable that we have called "value", it is the target value of the quantity useful for the regulation of the buck. For example, CV regulation will take a voltage value, CC regulation will take a current value. In the case of MPPT regulation, we take no argument because this regulation aims to send as much power as we can from the photovoltaic panel therefore no need to have a target value because it is the maximum power extracted from the panel that will be taken.

5. Use case example

Here, we are going to illustrate the use of these functions and variable through an concrete example.
We will take as an example, the system in CV regulation mode, the costumer wants to change the regulation mode and switch to MPPT regulation, he will therefore interact with the system to be able to change the current regulation. The client will dialogue with external part of the system, he will ask the Set_Mode_MPPT system without giving any argument, we are currently in the right side of Figure 3.1. Following this function, the External_User_API will change the value of the global variable RegMode and will set it to the value which corresponds to the MPPT regulation thanks to the internal part will be able to read the new value of the global variable RegMode and change the regulation of the buck through the regulateMPPT function. We are now in the left side of Figure 3.1. Through this mechanism of external and internal part, we limit the interactions between the user and the microcontroller. In order to be able to have information on the changes taking place, the customer can use the different Get functions available to him.

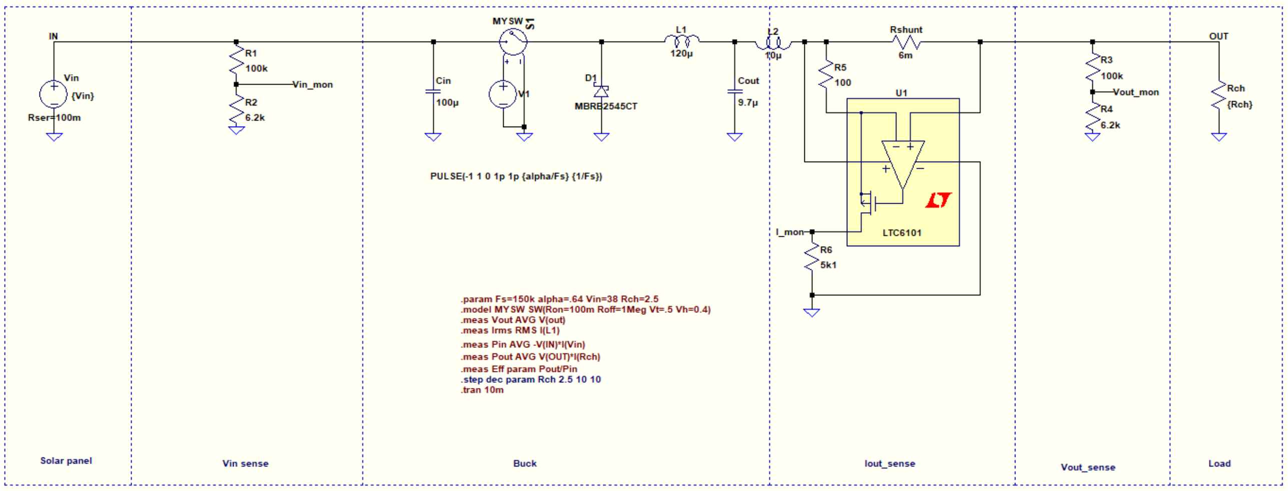
Chapter 4

Next Phase

As we have a complete low level architecture, we will now focus on the actual realisation of the project. The software development will be about the project library and also test functions. In parallel, we will pursue the PCB layout on Altium and make further simulations on LT Spice. In two weeks we will deliver the schematic and the BOM (Bill of Materials), which will show the complete circuit and all the components needed. Next month we will write a report with the final software architecture. We predict that we will put our boards into fabrication in two months. Our team will work effectively to properly reach those goals and satisfy the client wishes.

Appendix A

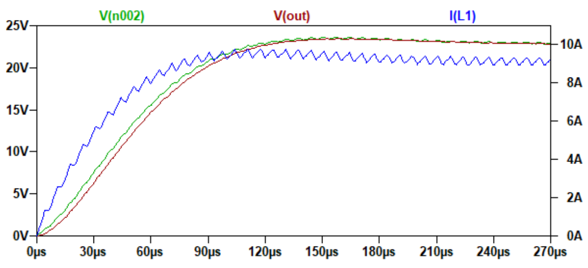
LTSpice simulation



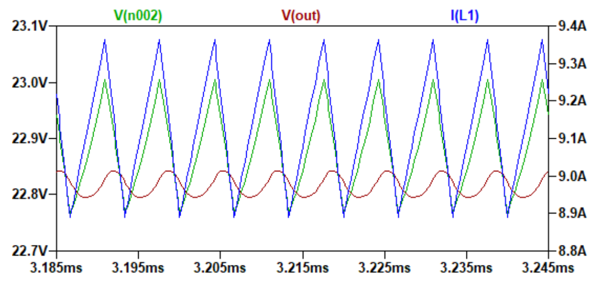
```
PULSE(-1 1 0 1p 1p {alpha/Fs} {1/Fs})
```

```
.param Fs=150k alpha=.64 Vin=38 Rch=2.5
.model MYSW SW(Ron=100m Roff=1Meg Vt=.5 Vh=0.4)
.meas Vout AVG V(out)
.meas Irms RMS I(L1)
.meas Pin AVG -V(IN)*I(Vin)
.meas Pout AVG V(OUT)*I(Rch)
.meas Eff param Pout/Pin
.step dec param Rch 2.5 10 10
.tran 10m
```

Figure A.1: LTSPICE schematic for simulation



(a) At startup



(b) Steady state

Figure A.2: Output Current & Voltages

$V(n002)$ is the voltage across Cout. Note the small current overshoot, and the improved voltage ripple at $V(out)$.