Chipcon

RadioDesk™ Reference Design and Protocol

User Manual

Rev. 1.3

Introduction

RadioDesk™ protocol

RadioDesk™ hardware reference design

RadioDesk™ firmware

Customization

Supported chips

- CC2500

Table of contents

Table of figures

1. Introduction

RadioDesk™ is Chipcon's platform for wireless HID (Human Interface Device) devices. This platform is based on Chipcon's SmartRF®04 radio ICs, and includes a complete reference design as well as an advanced RF adaptive frequency hopping protocol.

The RadioDesk™ reference design is a complete, production ready wireless desktop (keyboard and mouse) design. Focus is on low cost, a robust RF link in the face of heavy interference from other 2.4 GHz RF systems and long battery lifetime.

The RadioDesk™ protocol is an advanced adaptive frequency-hopping protocol and designed specifically for wireless HID use. It is more efficient and optimised for this purpose compared to other more general-purpose protocols. Interoperability between different implementations of RadioDesk™ has not been a priority, as this would compromise the main goals. Implementers are also free to modify RadioDesk™ to suit their specific purposes.

Note that RadioDesk™ is not a one-off project by Chipcon; Chipcon intends to continue to improve and optimise these materials. Make sure you are kept updated with the latest developments by subscribing to our Developer's Newsletter.

This advanced 2.4 GHz solution has many advantages compared to a conventional 27 MHz design. The most important are the improved range and the possibility of having many systems operating in a small area.

The RadioDesk™ reference design firmware source code as well as hardware Gerber files are available free-of-charge, please contact your local Chipcon representative to receive these files.

2. About this manual

This manual presents both the RadioDesk™ protocol and the RadioDesk™ reference design. It provides all the technical material needed for an implementer to duplicate and modify both the hardware and software. It also explains in detail how the design works, and possible ways of modifying it.

3. Definitions

HID	Human Interface Device – the term used in USB nomenclature covering keyboard, mice and other user-interface devices
TDMA	Time Division Multiple Access – a method for supporting multiple devices where devices coexist by transmitting at different times
PTU	Protocol Time Unit – the length of a time slot
CDMA	Code Division Multiple Access
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
CRC	Cyclic Redundancy Check
FSK	Frequency Shift Keying
MSK	Minimum Shift Keying
HAL	Hardware Abstraction Layer
MAC	Medium Access
RSSI	Received Signal Strength Indication

4. RadioDesk™ protocol

4.1 Overview

The RadioDesk™ protocol is an advanced protocol for wireless HID devices and includes adaptive frequency hopping. The protocol was based on the following design criteria:

- Coexistence with existing 2.4 GHz systems
- Use of ultra-low-cost MCUs
- Long battery life time
- Low latency

RadioDesk™ has been designed to be easily customizable to the implementer's requirements. It can support a variable number of peripherals, various data formats, both one-way and two-way RF links and so on.

RadioDesk™ is a master-slave protocol. In a standard wireless mouse or keyboard system, the USB or PS/2 dongle will be the master and the other devices will be slaves. To save power, the protocol has been designed for asymmetric power consumption. It is assumed that the master has a plentiful source of power, while the slaves are power-constrained. This holds true in most HID systems.

RadioDesk™ uses time-division multiplexing (TDMA) to support multiple devices. A big advantage of the TDMA approach is that it provides infinite attenuation between the various devices, unlike a CDMA or FDMA approach, where differing signal strengths can cause problems. A frame is divided into n+2 time slots, where n is the number of slaves in the system. In the first time slot, the master transmits a beacon. This beacon serves three purposes; it allows for data transfer from the master to the slaves, inform the slaves of the network status, and allow the slaves to synchronise their timing to the master.

After the beacon, the slaves transmit data to the master if they have data to report. The last time slot is used by the master to scan a channel for use in the frequency adaptivity.

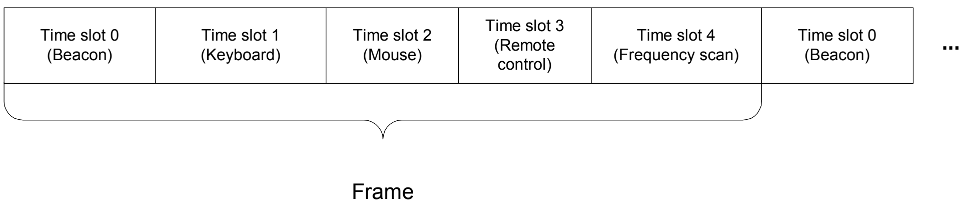


Figure 1. Basic RadioDesk™ frame structure

Figure 1 shows how this looks in a system with three slaves; a keyboard, a mouse and a remote control. Once a frame is complete, the next frame follows immediately afterwards.

The slaves can power-down at any time, they are not required to respond to the beacon unless they have data to report.

4.2 Terminology

As described above, the basic protocol period starting with a beacon and ending just before the next beacon is called a *frame*. Each frame is sent and transmitted on a new channel in the frequency hopping sequence.

A frame is divided into *time slots*. There is sufficient space in a time slot to transmit or receive a single packet. Each time slot is dedicated to a single unit. The time slots are per definition one PTU (Protocol Time Unit) long.

A *packet* contains data that is sent from one unit to another. It consists of a preamble, sync word and payload data followed by a CRC. The packet format is described in more detail later on.

A *channel* is a logical number applied to a physical frequency. RadioDesk uses up to 64 channels, numbered from 0 to 63, even though the physical frequencies used for each channel may vary depending on the implementation.

Active channels are the 4 channels that are active in the frequency-hopping scheme.

4.3 Physical layer

RadioDesk™ is designed to operate in the worldwide, 2.4 GHz license free frequency band, but could be ported to work at other frequencies. Since the CC1100 is register-compatible with the CC2500, it is very easy to port RadioDesk™ to work in the US 902-928 MHz frequency band, for instance. The implementer has to make sure that RadioDesk™ complies with appropriate regulatory requirements (for instance duty cycle or frequency hopping requirements).

RadioDesk™ is designed to operate with a RF data rate of 250 kbps. The RadioDesk™ reference design uses the CC2500 transceiver and utilizes MSK as modulation method. It is easy to modify the system to operate at different data rates. One very simple modification is to increase the data rate to 500 kbps.

RadioDesk™ can also be implemented on other RF transceivers. For instance, some implementers may want to utilize an IEEE 802.15.4 compatible radio such as the CC2420. The data rate is the same, but the number of total channels need to be reduced. Other than that, the software may be left essentially unchanged.

Parameter	Allowed range	Unit
Frequency range	2401-2464	MHz
Frequency spacing	1	MHz
Data rate	250 (500 if FEC is to be used)	kbps
Modulation	FSK or MSK	
PLL lock time / start time from IDLE (blanking period)	<192	µs
PTU	2000	µs
Spread spectrum technique	Adaptive frequency hopping	
Number of channels used	4 active, selected from 64 possibilities	

Table 1. Basic physical layer parameters

4.4 Frequency hopping

RadioDesk™ is an adaptive frequency hopping protocol. Both the master and the slaves change frequency for each frame.

RadioDesk™ uses four active frequencies at any one time. These are selected from a set of 64 possible channels. The master scans through all 64 channels continuously, at a rate of one channel each frame. If one of the active channels has significantly worse performance than the best non-active channel, then the master switches out the bad channel with the good one. To ensure that the active channels are not so close together that a single wide-band interferer could jam all of them at once, there is a limitation on the channel selection algorithm.

The frequency hopping is controlled by a 15-bit pseudo-random sequence. This sequence has a length of 32767 bits. Since 32767 is not divisible by 2, and 2 bits are used to determine the channel used, the RadioDesk™ hopping sequence will only repeat after 32767 hops have been made. The pseudo-random sequence can be implemented as a 15-bit shift register with the appropriate XOR feedback. By setting the shift register to a value, it is possible to select a position in the sequence. This is called the *seed* by analogy to random number generator algorithms. Two units can ensure that they are at the same place in the sequence by comparing the value of the entire shift register. The seed is transmitted in beacon packets so that slaves can synchronise with the pseudo-random sequence of the master.

The implementer might choose to use a purely sequential switch between the four active channels. This would make the two seed bytes in the beacon redundant.

4.5 Data flow

The master transmits a beacon packet in the first time slot of each frame. The beacon packet contains protocol status information and data which the master wants to communicate to the slaves. In a system where the master needs to transmit more data to the slaves than there is place for in the beacon packet, there are two possibilities: Either the time slot length can be extended or an extra time slot can be added for beacon transmission.

When a slave has data to transmit, it listens for a beacon and then transmits a packet in its dedicated time slot. To listen for beacons is not mandatory for a slave; it is free to go into power-down mode at any time to save power. In addition, the slaves do not need to transmit a packet if they do not have data to provide. If a slave transmits data to the master, the master will use the acknowledge bits in the next beacon to confirm the data transfer. If the appropriate acknowledge bit is not set, the slave can retransmit the packet in the next frame.

Figure 2 shows a typical RadioDesk™ usage scenario.

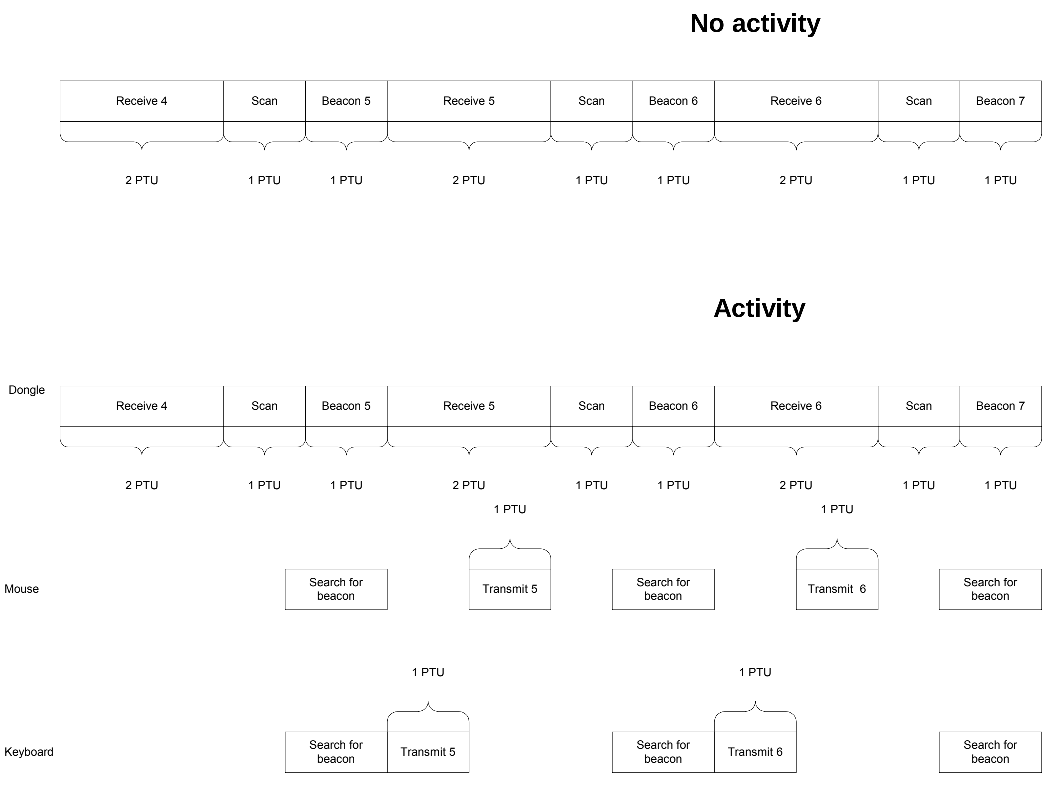


Figure 2. Typical RadioDesk™ data flow

4.6 Packet format

4.6.1 General packet format

RadioDesk™ employs a general packet format, but the packet format can be modified to suit the application if necessary.

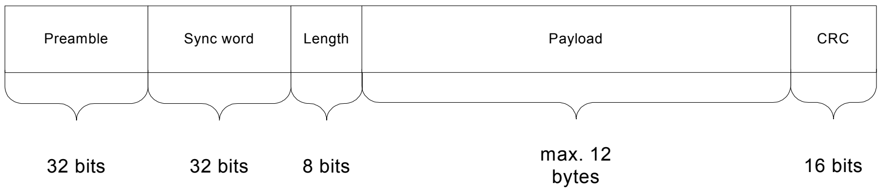


Figure 3. Basic packet format

The packet starts with a preamble, which is used to settle the receiver chain. Sync word is then sent to differentiate different systems. This is followed by a length byte which indicates the length of the payload field (excluding the CRC and the length byte itself). The last part of the packet is a 16-bit CRC used to check for packet corruption.

4.6.2 Beacon payload

The beacon contains 12 bytes of payload; the meaning of each field is shown in
Table 2 below.

Byte pos.	Name	Field	Bit pos.	Length	Purpose	Comments
0	Network ID			16 bits	Unique ID for each RadioDesk™ network	
2	Seq/type.	TYPE	7..5 4..0	3 bits 5 bits	Type of packet	000=Beacon Reserved for future use
3	Status	POWER SAVE	7	1 bits	Enable power saving in slaves	
		Reserved	6..2	5 bits	Not used	
		MOUSE bind request	1	1 bits	Indicate that a mouse may bind	
		KEYBOARD bindrequest	0	1 bits	Indicate that a keyboard may bind	
4	Hop sequence info			16 bits	Current position in hop sequence	
6	Ack			8 bits	Ack of slave messages previous frame	LSB=slave 0 MSB=slave 7 0: keyboard 1: mouse
7	Active channels					Byte pos 7: CH1 Byte pos 8: CH2 Byte pos 9:

		RESERVED FREQ	7..6 5..0	2 bits 6 bits	Not used Active frequency on channel	CH3 Byte pos 10: CH4
11	Data	Reserved	7..0	7 bits	Application-dependent, can be used to send data from master to slaves (Caps-lock status etc)	

Table 2. Beacon payload

4.6.3 Data payload

Byte pos.	Name	Field	Bit pos.	Length	Purpose	Comments
0	ID			16 bits	Unique ID for each RadioDesk™ network	
3	TYPE	TYPE RESYNC Sequence number	7..5 4 3..0	3 bits 1 bit 4 bits	Type of packet Set if device is waking up. Cleared by first ACK from dongle. Control retransmissions	001=Mouse 010=Keyboard
4	Status		7..0	1 bit	Paired bit	Set when a slave has been paired
4			6..0	7 bits	Implementation dependent	May be used to report, say, motion sensor status or number of characters in the keyboard buffer
5	Data			max. 6 bytes	Used to report data from slaves to master (mouse movements, keyboard hits etc.)	

Table 3. Data payload

4.6.4 Mouse data

The optical sensor typically provides two 8-bit delta values for X and Y movement. This is also what is required for boot functionality by the PC BIOS.

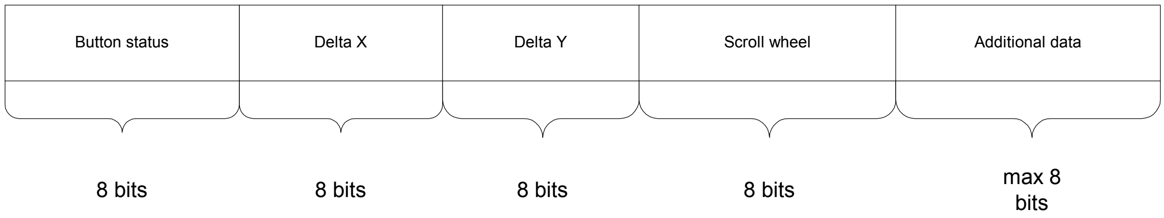


Figure 4 Suggested mouse data

In the additional data field it’s possible to send data such as battery status, scroll wheel data and so on. When buttons are pressed, the mouse should send data packets in each frame. The master should implement a keep-alive function, so that if no data is received from the mouse for several frames, the master should report to the PC that all buttons are released. This prevents stuck buttons if the radio link goes down.

4.6.5 Keyboard data

Boot functionality demanded by PC BIOS consists of the first byte of modifier keys, then up to 6 key codes.

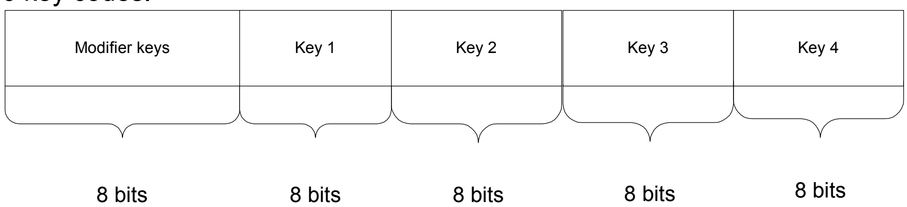


Figure 5 Suggested keyboard data

If LED indicators for caps-lock, num-lock and so on are included, data can be sent in the single data byte available in the beacon packet.

4.7 IDs and pairing

The RadioDesk™ reference design uses 16-bit IDs, but may easily be amended to use 24 bits. This serial number is embedded into the OTP MCU in the master device. In order to minimise the chance of collisions, a random ID should be selected before starting a manufacturing batch, and then the IDs should be incremented sequentially for each set in the batch. Randomizing the start number is wise; otherwise the first production units for each manufacturer would start at 0, and be likely to be equal to equipment from other manufacturers.

The likelihood of two different sets having the same ID can be found from the so-called Birthday Theorem in statistics (so-called as the archetypical use is to find out how likely it is that two persons in a group share the same birthday):

$$P(collision) < \frac{q(q-1)}{2N}$$

Equation 1. Birthday theorem

q is the number of units and N is the number of possible IDs.

Another approximation is that approximately one collision will occur if you have $\sqrt{N}$ units within range. This turns out to be 4048 units for a 24-bit ID, or 256 units for a 16-bit ID.

The following table gives the upper bound on the collision probability for 16-bit and 24-bit IDs (assuming a random distribution):

Number of units within range	**Upper bound on probability of collision (24-bit ID)**	**Upper bound on probability of collision (16-bit ID)**
1	0	0
2	5,96046E-08	1,52588E-05
4	3,57628E-07	9,15527E-05
8	1,66893E-06	0,000427246
16	7,15256E-06	0,001831055
32	2,95639E-05	0,007568359
64	0,000120163	0,030761719
128	0,000484467	0,124023438
256	0,001945496	0,498046875
512	0,007797241	1
1024	0,031219482	1

Table 4. Probability of ID collisions

As can be seen, the probability of a network collision is remote when using a 24-bit ID as long as the number of units within range is below several hundreds. So many units will never be within range of each other, even in a very crowded office environment. Problems due to network collisions will be less likely than problems due to hardware failures in the field. Using a 16-bit ID is only recommended where only a few units will ever be within range of each other.

In some cases such a system is not practical (for example when using ROM-based MCUs), and another form of pairing should be used. The conventional way of doing pairing is having a button on each of the units, and then require the user the press these buttons to pair the devices (the likelihood of another unit close-by being in pairing mode at the same time is very remote). This works well, but requires the user to perform this procedure every time the batteries are changed. The master contains a fixed ID, which the slaves bond to. This is done because the master will lose power every time the PC is switched off, while the slaves can rely on battery power.

RadioDesk™ implements a simple pairing scheme where the master has a fixed ID and the slaves attempt to bind to the dongle when they are reset. The slave will attach to the first encountered master sending a beacon that it receives with a signal strength above a certain threshold.

4.8 Power saving

RadioDesk™ allows the slaves to power-down at any time in order to save as much power as possible. When a slave wakes up, it first checks the channels that were active when it powered down. If it does not find the beacon in a predefined number of tries, it goes into search mode and searches all the 64 possible channels.

When searching, the slave hops between random frequencies. If no frequency changes have occurred since the slave went to sleep, Figure 6 shows the probability of finding the beacon after a certain number of frames.

If a single frequency has been changed, then the probability distribution in Figure 7 is correct.

If all four frequencies have been changed, or after a reset, then the probability distribution follows the curve in Figure 8.

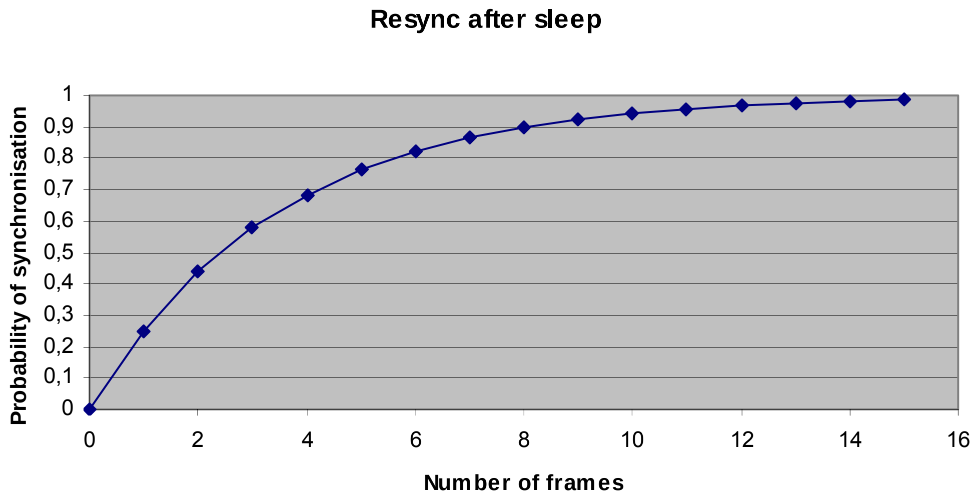


Figure 6. Resynchronisation after sleep

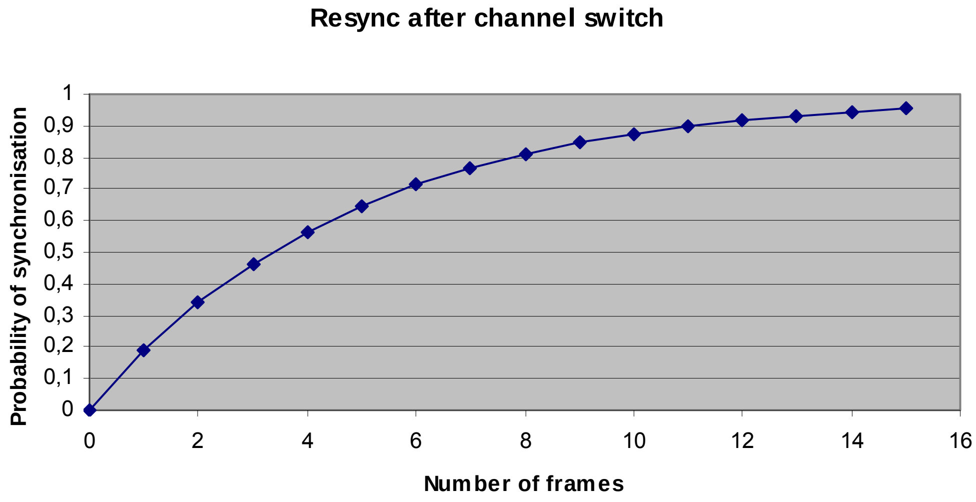


Figure 7. Resynchronisation after channel switch

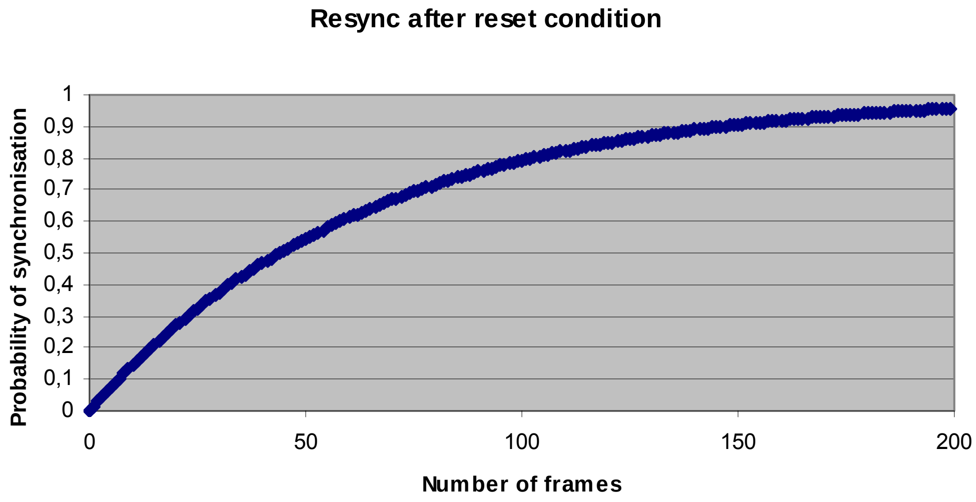


Figure 8. Resynchronisation after reset

Typical HID devices will be used erratically, with periods of use being separated by long periods of inactivity. Studies show that the user can tolerate somewhat longer latency for the first event after a period of inactivity. The device should therefore look at the time since the last event, and switch into a low-power mode after a certain period of inactivity. This approach has been adopted by the manufacturers of movement sensors and is used in optical mice.

4.9 Battery life time

Battery lifetime is one of the key parameters for a wireless HID solution. In order to estimate this, it is necessary to know the characteristics of the battery that will be used as well as knowing how the device will be used.

In many aspects, alkaline AA or AAA cells are ideal for powering a HID device. They are inexpensive and available just about everywhere. Good quality AA cells have a rated capacity of 2850 mAh, while AAA cells are rated to 1250 mAh. End-of-life voltage is usually given to be 0.9 V.

Depending on the MCU and other circuitry used, several voltage regulation schemes are possible:

1. The unit may be powered directly from the batteries. The CC2500/CC2550 radio supports a voltage supply range of 1.8 – 3.6 V. This fits very well with the 1.8 – 3.2 V range of two serially connected alkaline cells. If the other circuitry in the unit can also operate with this input voltage range, then this is the preferred option, as it provides the lowest cost and utilizes the battery capacity to the best extent. It is possible to use direct powering even if the minimum supply voltage of the MCU or another critical component is more than 1.8V, but then the battery life time will be reduced accordingly.
2. The unit is powered from 4 alkaline cells, using a 3 V or 3.3 V linear regulator. This will allow the unit to utilize the whole battery capacity, but 4 batteries are physically bulky. This can be a good solution for a wireless keyboard, as space is generally not a problem, but is not practical for a wireless mouse because of size and bulk. Some capacity is wasted in the voltage regulator.
3. The unit is powered from 1 or 2 alkaline cells, using a switch-mode boost converter. This may be required if other circuitry requires a 3.3V supply (for example the optical sensor in a wireless mouse), and size constraints forbid the use of more than 2 cells. This increases cost (~ USD $0.30 in large quantities) and some battery capacity is wasted because the voltage regulator efficiency is never 100%, but it does work down to the minimum battery voltage and enables very small designs using a single cell.

In order to estimate battery lifetime, the easiest technique is to calculate an average current draw, and then divide the capacity by the average current to find the battery life time in hours. Please note that battery capacity will be slightly lower when a varying current is drawn than if the current was constant.

To calculate an average current, it is necessary to calculate the length of time spent in various states and how much current is drawn in each state.

In order to simplify checking different scenarios for power consumption, Chipcon has made Excel spreadsheets that perform the appropriate calculations. These spreadsheets are enclosed in a .ZIP file together with this document.

4.9.1 Optical/laser mouse

An optical or laser mouse is typically the most power-intensive user-interface device because of the current consumption of the sensor. A LED is also used to provide light for the sensor. The mouse will therefore usually be the most power-constrained device.

Agilent has provided the following usage profile for a typical business user (home users usually use their systems much less, and will get correspondingly longer battery life times):

1.4% of active usage (with minimum latency)
4.0% of inactive usage (with higher latency on wake-up)
94.6% power-down (with the highest latency on wake-up)

The mouse supports four different power modes:

- Active, where the optical sensor is turned on all the time, the scroll wheel is active, the MCU is active and the radio is receiving, listening to beacons and transmitting position data to the master. If no activity is detected for 800 milliseconds, the mouse goes into semi-active mode.
- Semi-active, where the optical sensor is being duty-cycled to save power, the scroll wheel is not active, the MCU is active and the radio is listening to beacons. If activity is detected, the mouse goes to active mode. If no activity is detected for 60 seconds while in semi-active mode, the mouse goes into sleep mode.
- Sleep, where the optical sensor is being duty-cycled at a low rate, the scroll wheel is not active, the MCU is in sleep most of the time and the radio polls for beacons at regular intervals. If activity is detected, the mouse goes into active mode. If beacons have not been detected for 400 milliseconds the mouse enters deep sleep mode.
- When no beacons have been found for 400 milliseconds, then the PC is either shut off, out of range or in suspend mode. The mouse then enters deep sleep. In this mode, the optical sensor is shut entirely off, the scroll wheel is shut off, the MCU is in power down mode and the radio shut off. If the user presses a mouse button, the mouse first tries a sends wake-up messages and tries to wake the PC from suspend mode. If it is not successful, the mouse gives up and resumes deep sleep.

Figure 9 shows what events trigger transitions between the different power modes.

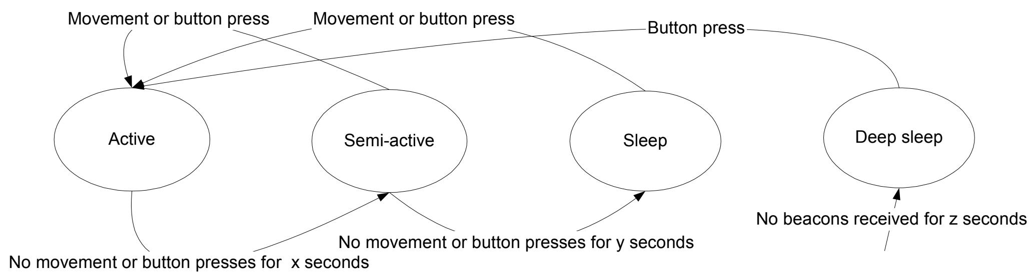


Figure 9. Optical mouse power mode transitions

In active mode, the MCU is active, the scroll wheel is active, the optical sensor is continually on and the radio is listening for beacons and transmitting data back to the master. All parts of the mouse except the radio are drawing maximum current the whole time. The radio is in RX mode for one time slot (1/4th of the total 8 ms frame duration for a mouse/keyboard combo), and in TX mode for the time it takes to send one packet (assuming a 20-byte packet, this is 0.64 ms). The radio will be in IDLE (synthesizer running) when it is not transmitting or receiving.

In semi-active mode, the MCU is active, the scroll wheel is turned off, the optical sensor is being duty-cycled and the radio is listening to beacon packets, being in IDLE for the rest of the time.

In sleep mode, the MCU is waking up at intervals, the scroll wheel is turned off, the optical sensor is being duty-cycled and the radio is being woken up periodically.

The radio wakes up every 5 seconds to check for beacons and frequency changes. When it wakes up, it needs to resynchronise. 90% of the time, this will take less than 8 frames.

In deep-sleep, the slave is completely off and the unit has to be woken up by a button press. It will then scan for beacon, and afterwards try to wake up the master if no beacon is found.

Please see the Excel spreadsheet for detailed calculations on power consumption.

4.9.2 Keyboard

The keyboard is usually less power-intensive than the mouse, because the system need only wake up when a key press is detected. Again, the usage pattern has a great deal of influence on the battery life time.

Since the CC2500 supports a voltage supply range of 1.8-3.6V, and the MCU used supports a voltage supply down to 2.2V, it is attractive to run all the circuitry directly from the batteries to save cost. It is estimated that the system can utilise 80% of the total battery life time if it cuts off at 2.2V.

In heavy business usage, the keyboard might be used 4 hours a day for 5 days a week. During this time, keyboard usage is irregular, but the total amount of time that the radio is active will be 10% of the time (the keyboard goes from active to sleep after 1 seconds of non-activity). This gives a total active time of around 20 minutes per day.

In active mode, the power consumption of the MCU is 0.7 mA, while the radio average current is 6.8 mA (same calculations as for the optical mouse in active mode). This gives a total average current of 7.5 mA. 20 minutes per day is equal to a 1.25% duty cycle. This gives an average current of around 0.1 mA. Sleep-mode current is approximately the same, assuming that the radio and MCU are active for 64 ms every 5 seconds to maintain information about frequency changes and so on. This gives a total average current consumption of 0.2 mA.

Please see the Excel spreadsheet for detailed calculations.

4.9.3 USB dongle

The USB standard mandates that any USB device that can wake up the computer should not draw more than 0.5 mA when the PC is in the low-power suspend state. To achieve this, the RF transceiver in the dongle must be power cycled (usually it is on all the time as in active mode the USB current limitation is 100 mA).

Assuming the USB MCU and mandatory USB pull-up resistors draw 0.25 mA, this leaves 0.25 mA or 250 uA for the radio. Since the CC2500 power-down current with wake-on-radio enabled is only 1.5 uA, this can be ignored in the calculation. Taking the worst-case CC2500 current consumption in RX of 15.6 mA into account, the duty cycle must be lower than 0.25/15.6, or 1.6%. This means that the CC2500 can be active for 16 ms every second. For the specific purpose of waking up a PC from suspend mode, it is suggested that a single-channel approach is taken. The reason for this is that long latency is usually acceptable, it is not a problem in the wake-up scenario, since wake-up itself takes many seconds. Secondly, most interferers are only transient in nature, so a device jamming the wake-up frequency for several seconds is not likely.

Wake-up functionality would not be possible at all using a low-cost MCU if it were not for the CC2500 wake-on-radio function, which enables the MCU to stay in the lowest-power state while the radio scans for activity. It is recommended that the wake-up channel be selected based on the network ID.

If wakeup is used, the keyboard and mouse will be programmed to transmit a long (several second) series of wake-up messages on the wake-up channel when user action occurs and the mouse and keyboard cannot find a beacon.

5. The RadioDesk™ reference design hardware

The RadioDesk™ reference design consists of a wireless mouse, wireless keyboard and a small form-factor USB dongle. The hardware has been designed with very low-cost components and with large-scale manufacturability in mind.

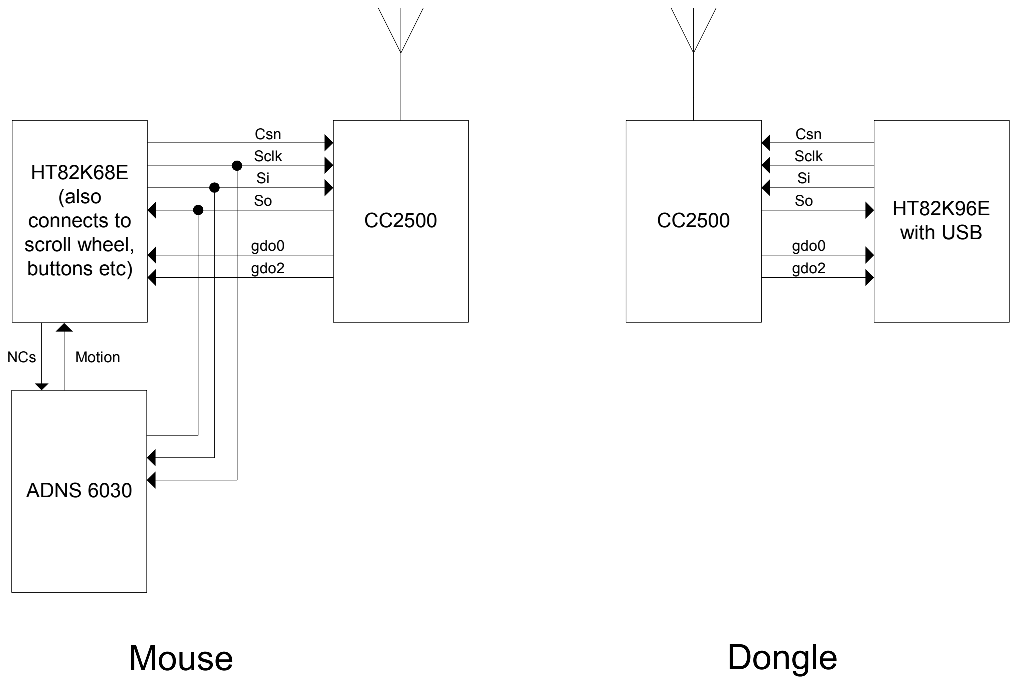


Figure 10. Example of RadioDesk system with dongle and laser mouse (Note: gdo0 is not used in RadioDesk firmware, but may be used in the future)

5.1 Small-factor USB dongle

The USB dongle has been designed to exhibit a minimum form factor. To achieve this, a 4-layer PCB has been used as well as a chip antenna solution.

A HTK96E low-speed USB MCU from Holtek Semiconductor was selected for the dongle. This MCU is powered directly from USB bus power. A 12 MHz crystal is used as clock source. The USB MCU communicates with the CC2500 transceiver using a bit-banged SPI serial interface, and reports data received from the keyboard and the mouse to the PC. A LED is included to show the link status.

The CC2500 2.4 GHz RF transceiver uses a balun circuit to interface with the single-ended chip antenna. The chip antenna is an advanced design from Fractus, which provides excellent antenna performance even though the size is very small. The CC2500 is powered from a 3.3V linear regulator.

The RF circuitry, the antenna and the voltage regulator are placed on the top side of the PCB, while the USB MCU is mounted on the bottom side. Actually, these directions are reversed with regards to the usual rotation of the USB plug, so the LED is mounted on the same side as the MCU as to be visible when plugged into a horizontally oriented USB socket. When laying out your own PCB, it is very important to copy the PCB layout of the RF section exactly. Layer 2 is used for a ground plane, while layer 3 is used to route power.

If price is more important than physical size, then it is possible to use copy the RF section and PCB antenna from the keyboard or mouse and connect this to the USB MCU and other circuitry in the dongle design. In this case, a 2-layer PCB can be used. The RF layout should be copied exactly from the keyboard or mouse, paying attention to ground planes as well. The usual alternative to a small USB dongle is to make a “hockey puck” formed device that connects to the PC via a USB cable.

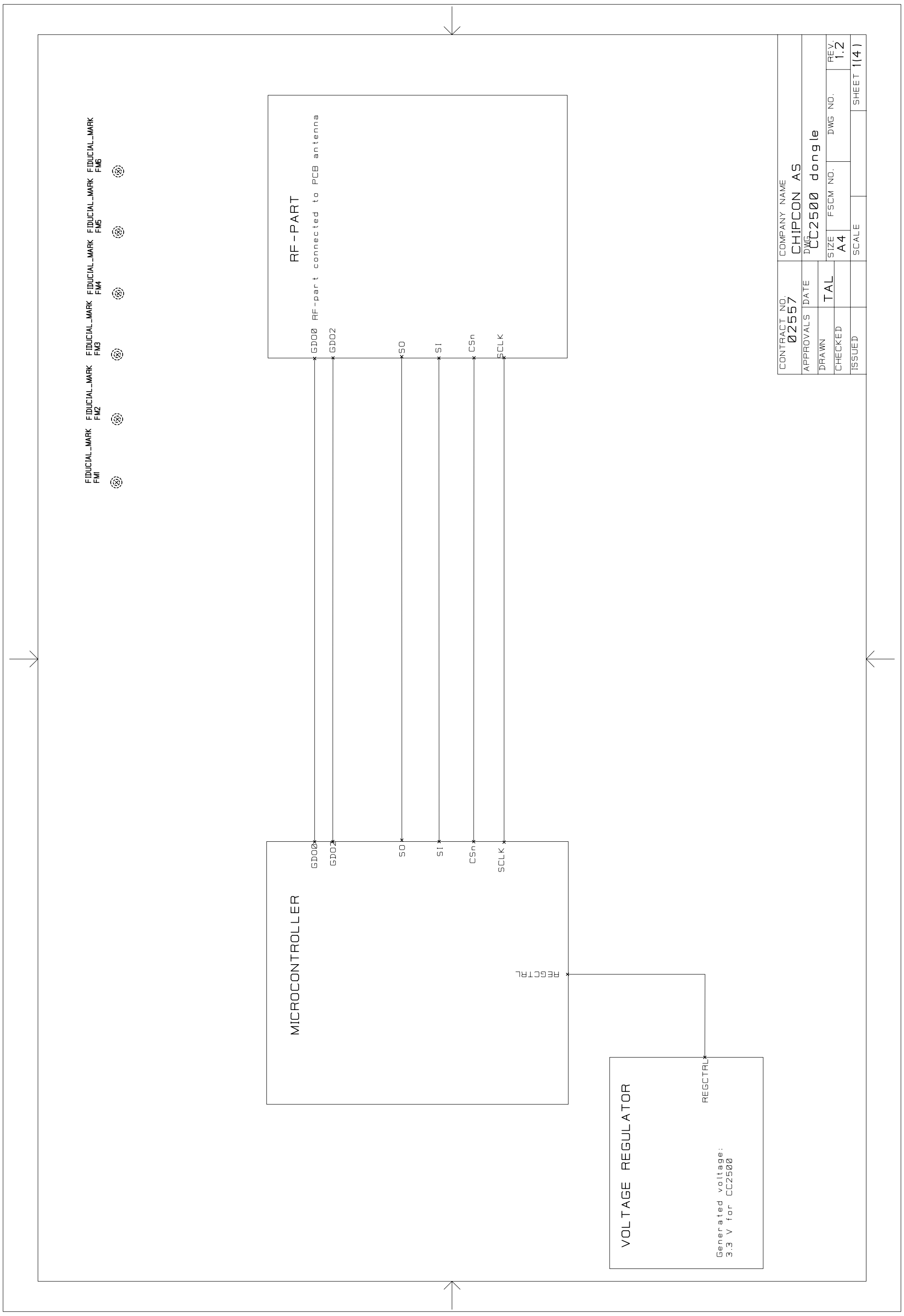


Figure 11. USB dongle schematic, page 1

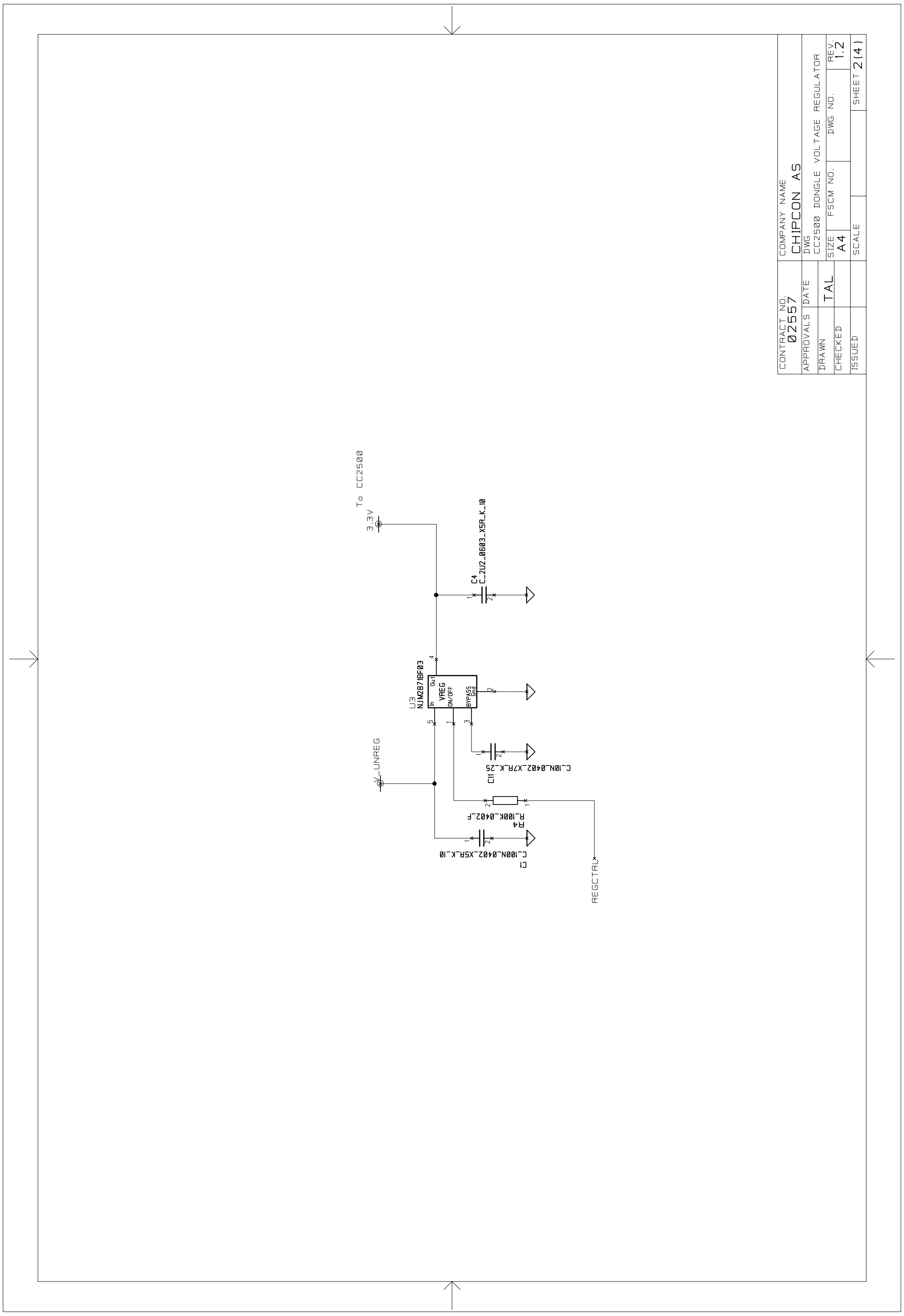


Figure 12. USB dongle schematic, page 2

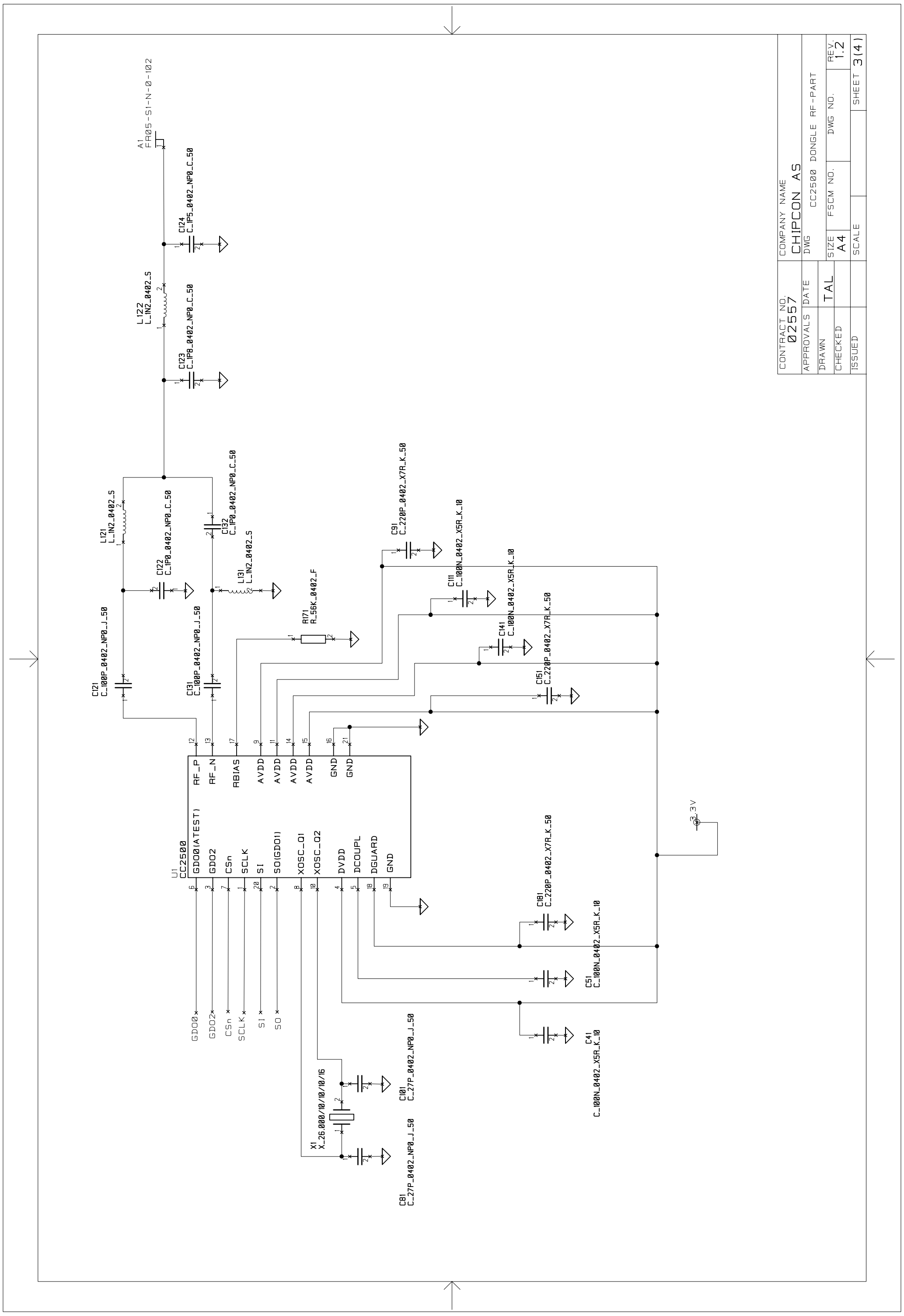


Figure 13. USB dongle schematic, page 3

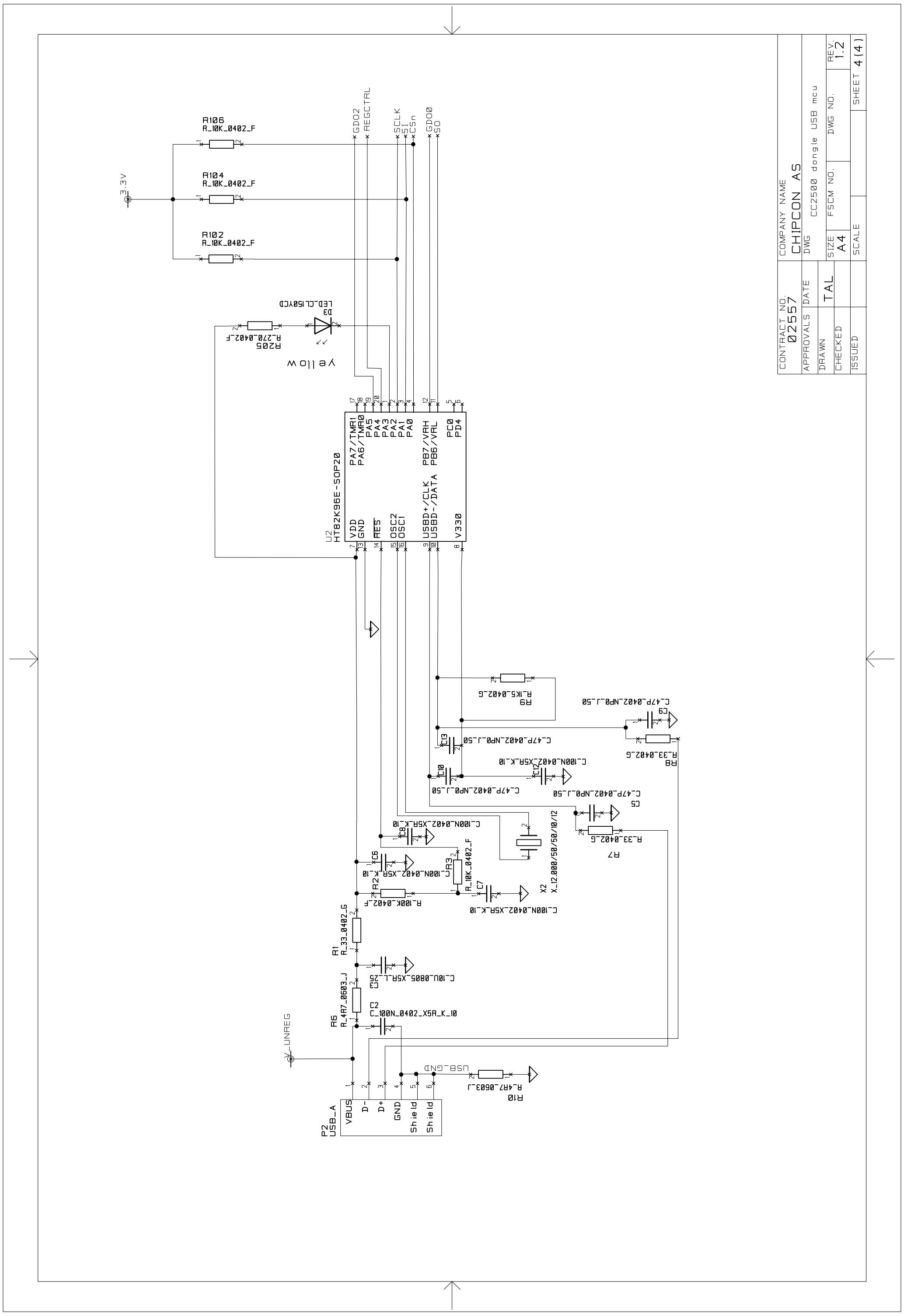


Figure 14. USB dongle schematic, page 4

Reference	Value/part number	Description	Manufacturer
A1	Compact Reach Xtend	Chip antenna	Fractus
C1	100 nF, X5R, 10%	Capacitor 0402	
C2	100 nF, X5R, 10%	Capacitor 0402	
C3	10 µF, X5R, 15%	Capacitor 0805	
C4	2.2 µF, X5R, 10%	Capacitor 0603	
C5	47 pF, NP0, 5%	Capacitor 0402	
C6	100 nF, X5R, 10%	Capacitor 0402	
C7	100 nF, X5R, 10%	Capacitor 0402	
C8	100 nF, X5R, 10%	Capacitor 0402	
C9	47 pF, NP0, 5%	Capacitor 0402	
C10	47 pF, NP0, 5%	Capacitor 0402	
C11	10 nF, X7R, 10%	Capacitor 0402	
C12	100 nF, X5R, 10%	Capacitor 0402	
C13	47 pF, NP0, 5%	Capacitor 0402	
C41	100 nF, X5R, 10%	Capacitor 0402	
C51	100 nF, X5R, 10%	Capacitor 0402	
C81	27 pF, NP0, 5%	Capacitor 0402	
C91	220 pF, X7R, 10%	Capacitor 0402	
C101	27 pF, NP0, 5%	Capacitor 0402	
C111	100 nF, X5R, 10%	Capacitor 0402	
C121	100 pF, NP0, 5%	Capacitor 0402	
C122	1.0 pF, NP0, ± 0.25 pF	Capacitor 0402	
C123	1.8 pF, NP0, ± 0.25 pF	Capacitor 0402	
C124	1.5 pF, NP0, ± 0.25 pF	Capacitor 0402	
C131	100 pF, NP0, 5%	Capacitor 0402	
C132	1.0 pF, NP0, ± 0.25 pF	Capacitor 0402	
C141	100 nF, X5R, 10%	Capacitor 0402	
C151	220 pF, X7R, 10%	Capacitor 0402	
C181	220 pF, X7R, 10%	Capacitor 0402	
D3	CL150YCD	LED, yellow, SMD	
L121	1.2 nH, ± 0.3 nH	Multi-layer inductor 0402	
L122	1.2 nH, ± 0.3 nH	Inductor 0402	
L131	1.2 nH, ± 0.3 nH	Inductor 0402	
P1	CONN10_MALE	Connector, 0.9mm pin, male (only for test)	
P2		USB plug type A connector	
P3	CONN10_MALE	Connector, 0.9mm pin, male (only for test)	
R1	33 Ω, 2%	Resistor 0402	
R2	100 kΩ, 2%	Resistor 0402	
R3	10 kΩ, 2%	Resistor 0402	
R4	100 kΩ, 2%	Resistor 0402	
R6	4.7 Ω, 5%	Resistor 0603	
R7	33 Ω, 2%	Resistor 0402	
R8	33 Ω, 2%	Resistor 0402	
R9	1.5 kΩ, 2%	Resistor 0402	
R10	4.7 Ω, 5%	Resistor 0603	
R102	10 kΩ, 2%	Resistor 0402	
R104	10 kΩ, 2%	Resistor 0402	
R106	10 kΩ, 2%	Resistor 0402	
R171	56 kΩ, 1%	Resistor 0402	

R205	270 Ω, 2%	Resistor 0402	
U1	CC2500	CC2500 2.4 GHz RF transceiver	Chipcon
U2	HT82K96E-SOP20	USB MCU	Holtek Semiconductor
U3	NJM2871BF03	3.0V LDO regulator	New Japan Radio
X1	X_26.000/20/20/10/16	26.000 MHz crystal, 20/20 ppm tolerance, HC-49/NX5032-SMD	
X2	X_12.000/50/50/10/12	12.000 MHz crystal, 50/50 ppm tolerance, HC-49-SMD	

Table 5. USB dongle bill of materials

5.2 Wireless keyboard

The wireless keyboard electronics consists of a small PCB which connects to the main keyboard matrix using carbon contacts. All the electronics are powered directly from the batteries. A HT82K68E serves as the system MCU, reading the keyboard matrix and running the radio protocol.

The CC2500 2.4 GHz RF transceiver uses a novel differential PCB antenna. This antenna has been designed to match the CC2500 RF I/O differential impedance, avoiding the need for any matching components or balun circuit. This ensures an absolute minimum number of external components, the CC2500 only requiring a single biasing resistor, a 26 MHz crystal, two crystal loading capacitors as well as decoupling capacitors.

Three LEDs have been included to indicate caps lock, num lock and scroll lock status. These may be illuminated only when the user types on the keyboard in order to save power, or even replaced with a custom LCD display (the LEDs are not implemented in RadioDesk firmware because it would require drilling holes in the keyboard chassis to make them visible).

The keyboard design does not support extended functionality like a scroll wheel and additional keys, but this can very easily be added.

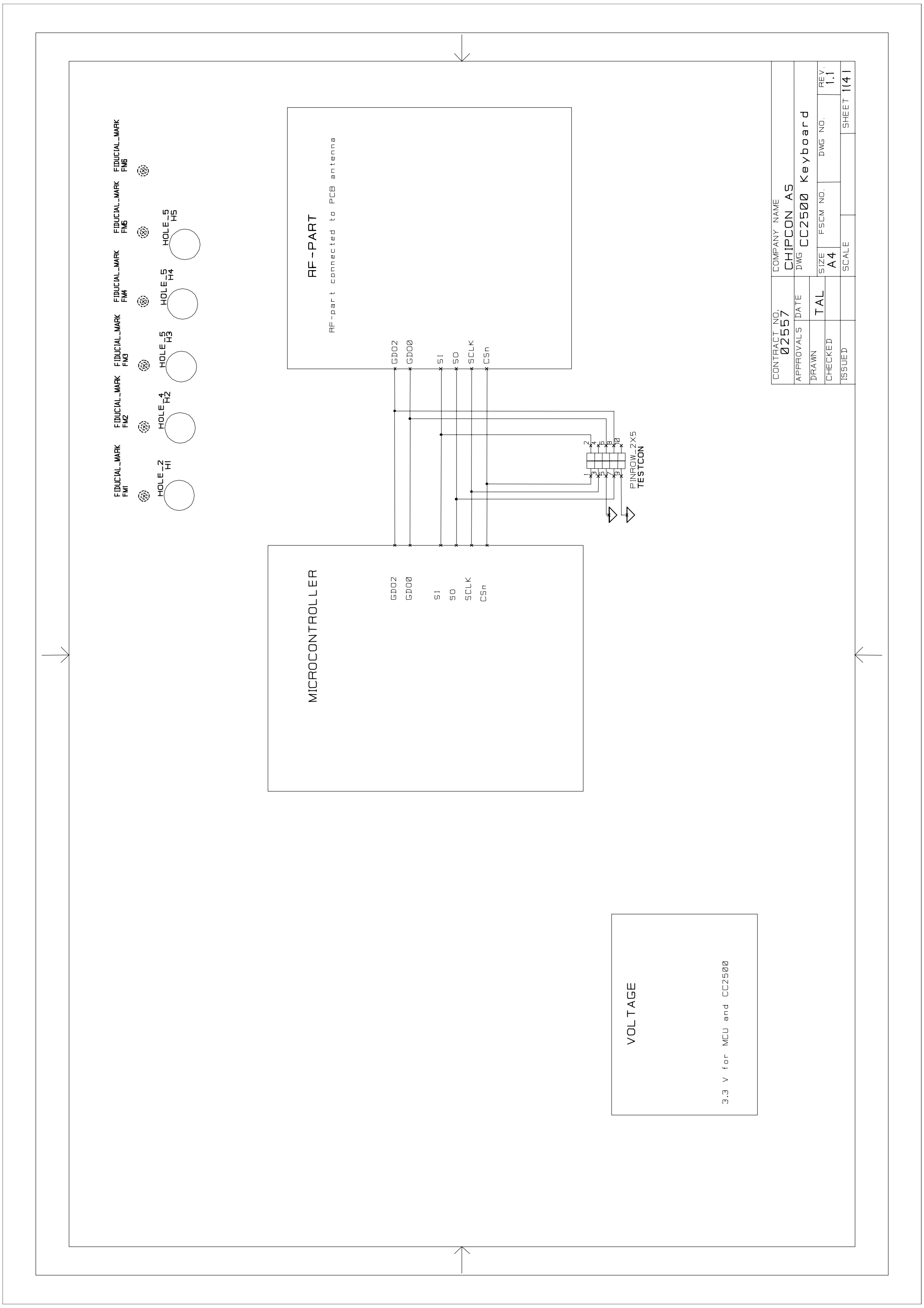


Figure 15. Keyboard schematic, page 1

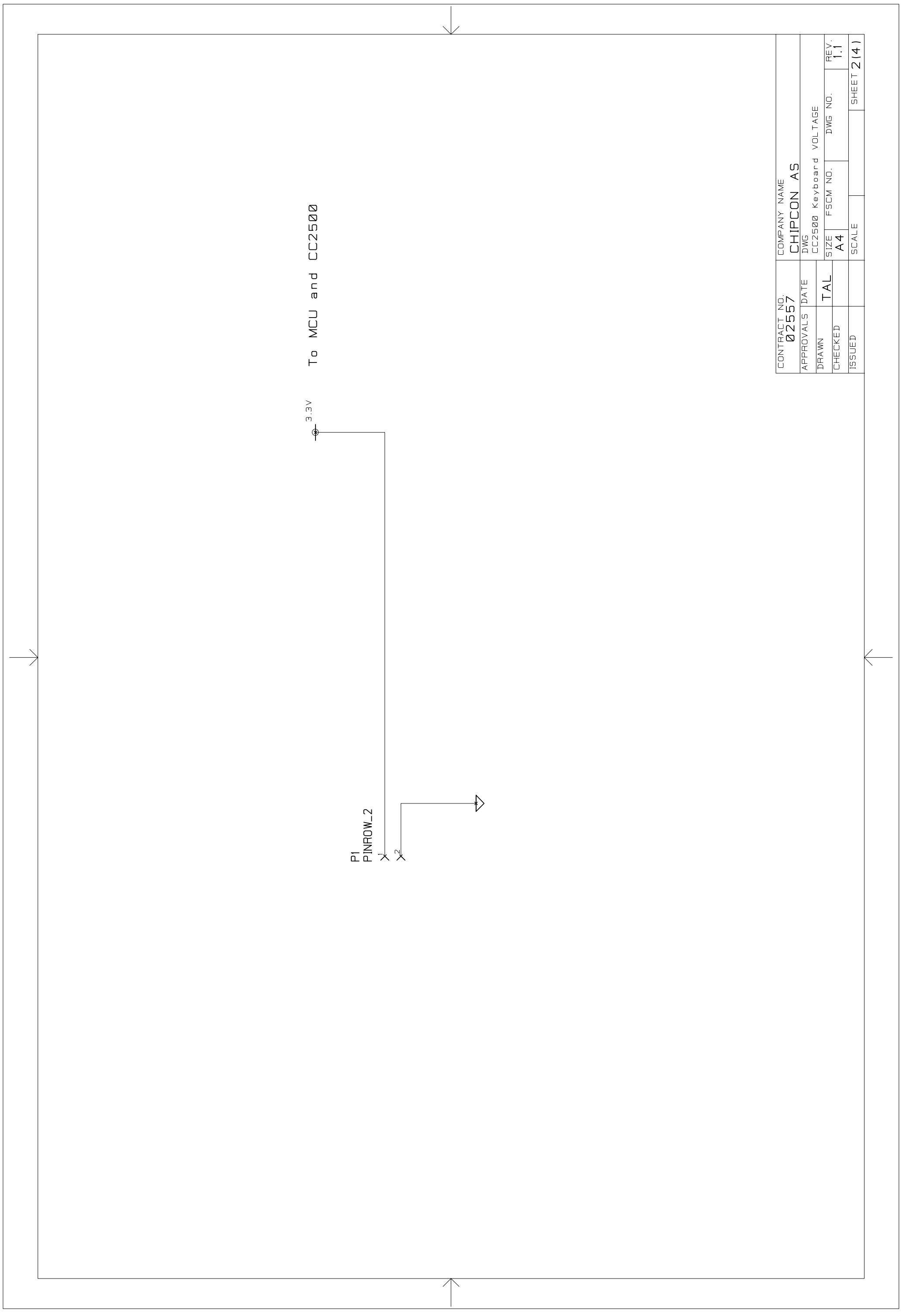


Figure 16. Keyboard schematic, page 2

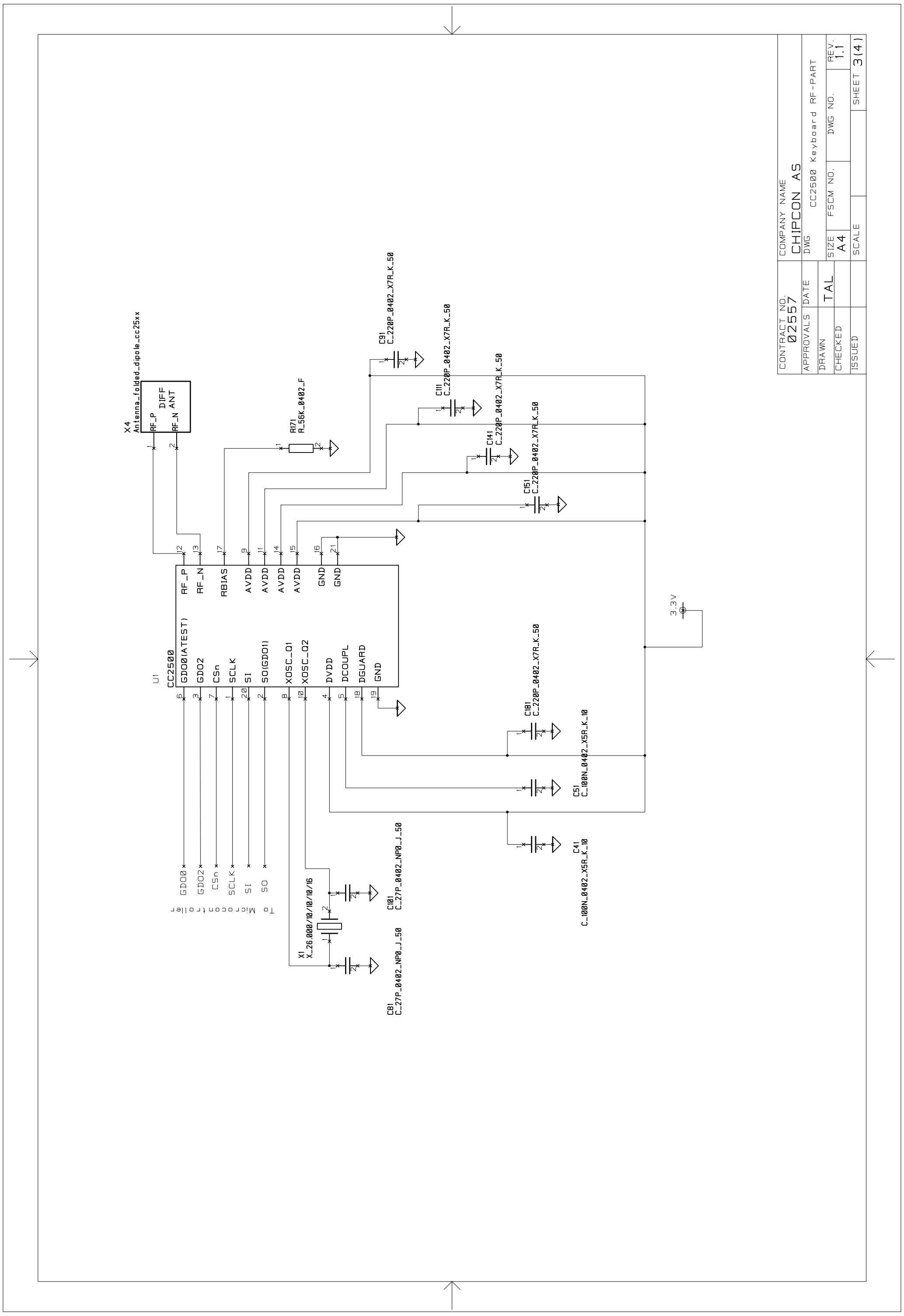


Figure 17. Keyboard schematic, page 3

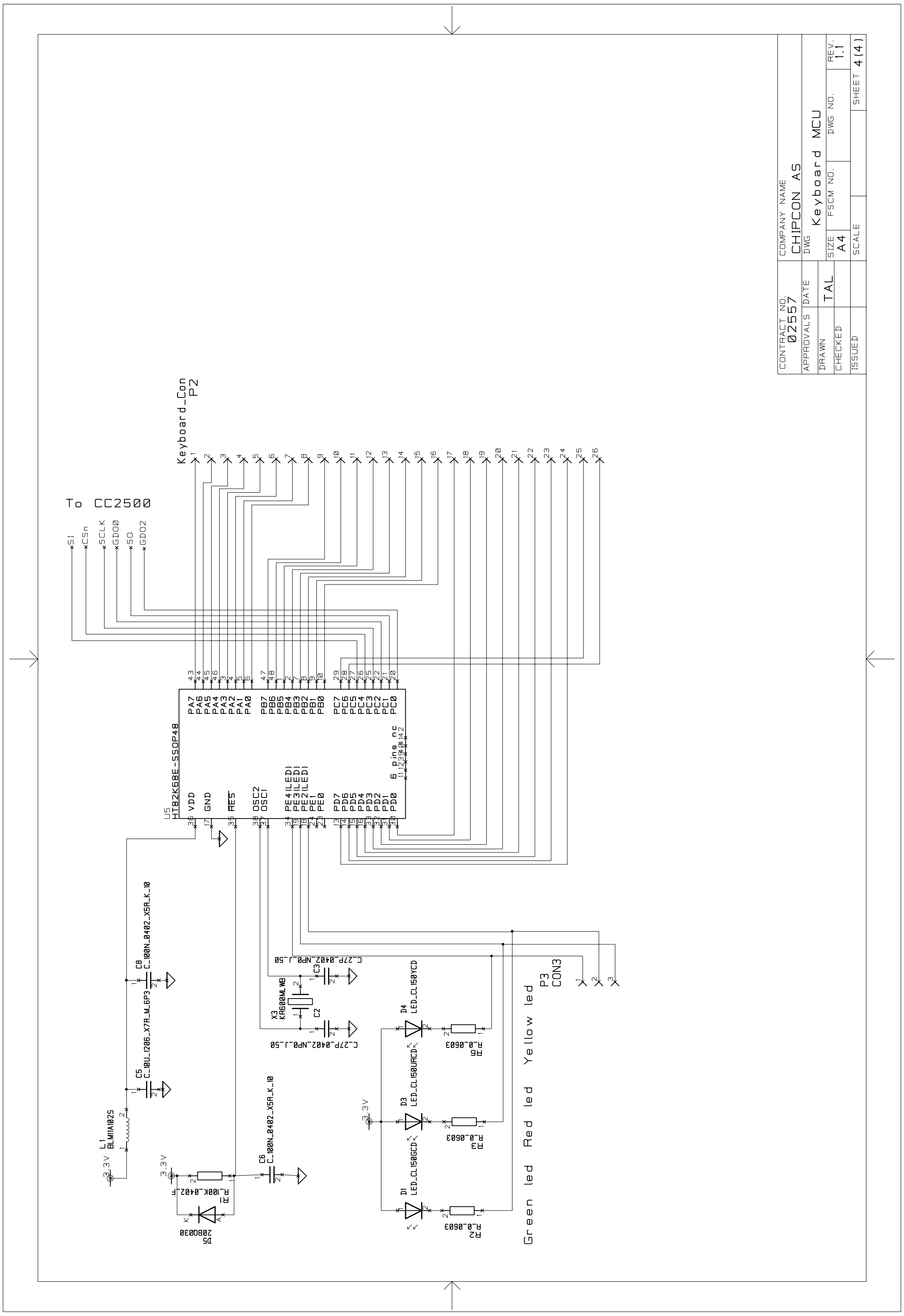


Figure 18. Keyboard schematic, page 4

Reference	Value/part number	Description	Manufacturer
C2	27 pF, NP0, 5%	Capacitor 0402	
C3	27 pF, NP0, 5%	Capacitor 0402	
C5	10 µF, X7R, 20%	Capacitor 1206	
C6	100 nF, X5R, 10%	Capacitor 0402	
C8	100 nF, X5R, 10%	Capacitor 0402	
C41	100 nF, X5R, 10%	Capacitor 0402	
C51	100 nF, X5R, 10%	Capacitor 0402	
C81	27 pF, NP0, 5%	Capacitor 0402	
C91	220 pF, X7R, 10%	Capacitor 0402	
C101	27 pF, NP0, 5%	Capacitor 0402	
C111	220 pF, X7R, 10%	Capacitor 0402	
C141	220 pF, X7R, 10%	Capacitor 0402	
C151	220 pF, X7R, 10%	Capacitor 0402	
C181	220 pF, X7R, 10%	Capacitor 0402	
D1	CL150GCD	LED, green, SMD	
D3	CL150URCD	LED, red, SMD	
D4	CL150YCD	LED, yellow, SMD	
D5	20BQ030	Schottkey diode	
L1	BLM11A102S	EMI filter bead	Murata
P1	PINROW_2	Pinrow, 2	
P3	CON3	Connector, 0.9 mm pin	
R1	100 kΩ, 2%	Resistor 0402	
R2	0 Ω	Resistor 0603	
R3	0 Ω	Resistor 0603	
R6	0 Ω	Resistor 0603	
R171	59 kΩ, 1%	Resistor 0402	
TESTCON	PINROW_2X5	Pinrow, 2x5 (for test)	
U1	CC2500	2.4 GHz RF transceiver	Chipcon
U5	HT82K68E-SSOP48	Keyboard MCU, SSOP48	Holtek
X1	X_26.000/20/20/10/16	26.000 MHz crystal, 20/20 ppm tolerance, HC-49/NX5032-SMD	
X3	KR600MLWB	6MHz resonator	Kingstate

Table 6. Keyboard bill of materials

5.3 Wireless mouse

The wireless mouse is implemented on two PCBs. The lower PCB includes the micro switches for the buttons, scroll wheel and optical sensor, while the upper PCB contains the rest of the electronics.

The electronics are powered from a TPS61078 switch-mode regulator from Texas Instruments. The reason for this is that the Agilent ADNS-6030 laser sensor needs a supply voltage between 3.0 and 3.6 V.

The same antenna and RF solution is used in the keyboard and the mouse.

The Z-wheel or scroll-wheel is of an optical type where a LED shines through the wheel itself, presenting a quadrature signal that is detected by two photo-transistors. The mouse buttons are implemented using standard micro-switches.

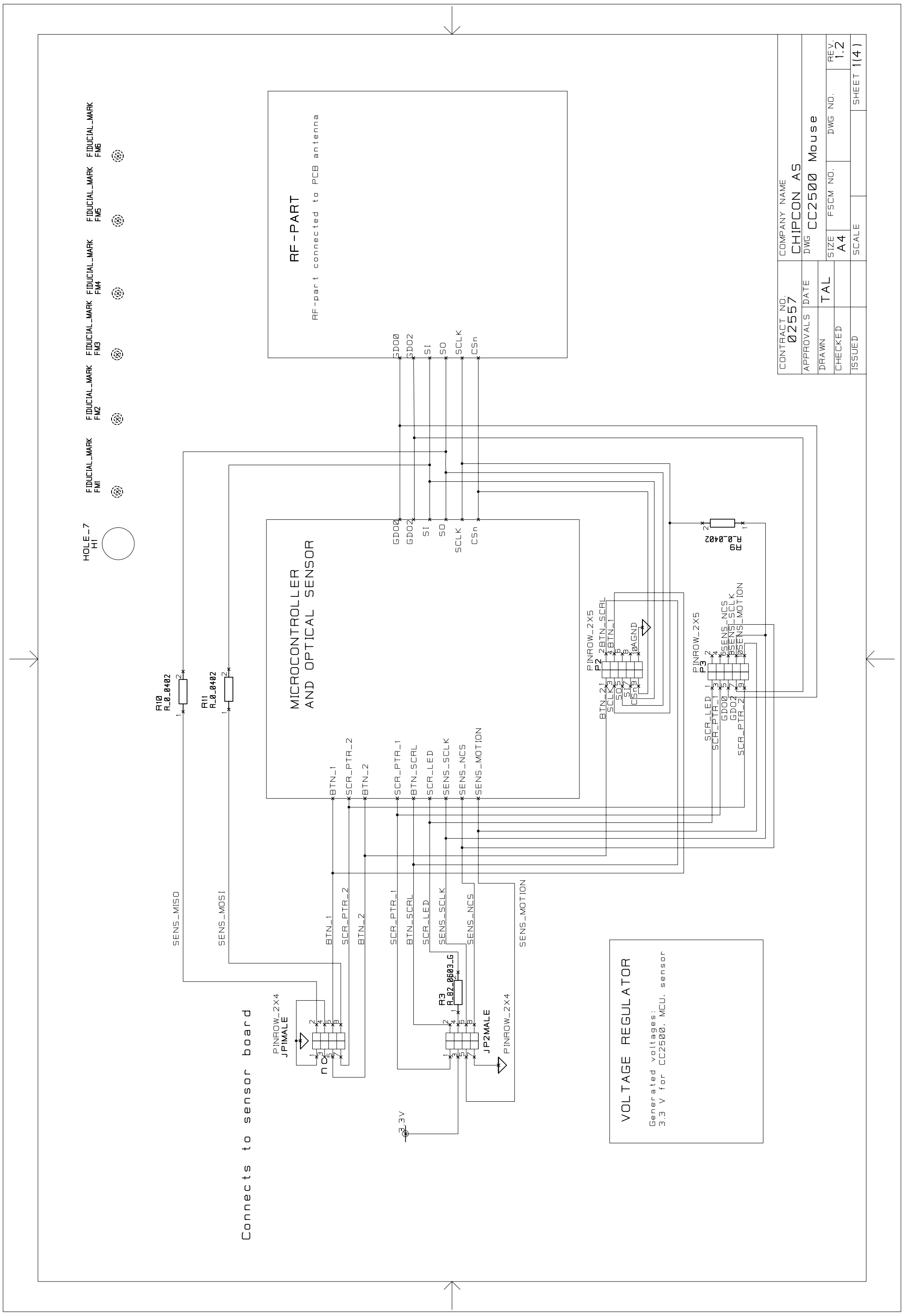


Figure 19. Mouse schematic, page 1

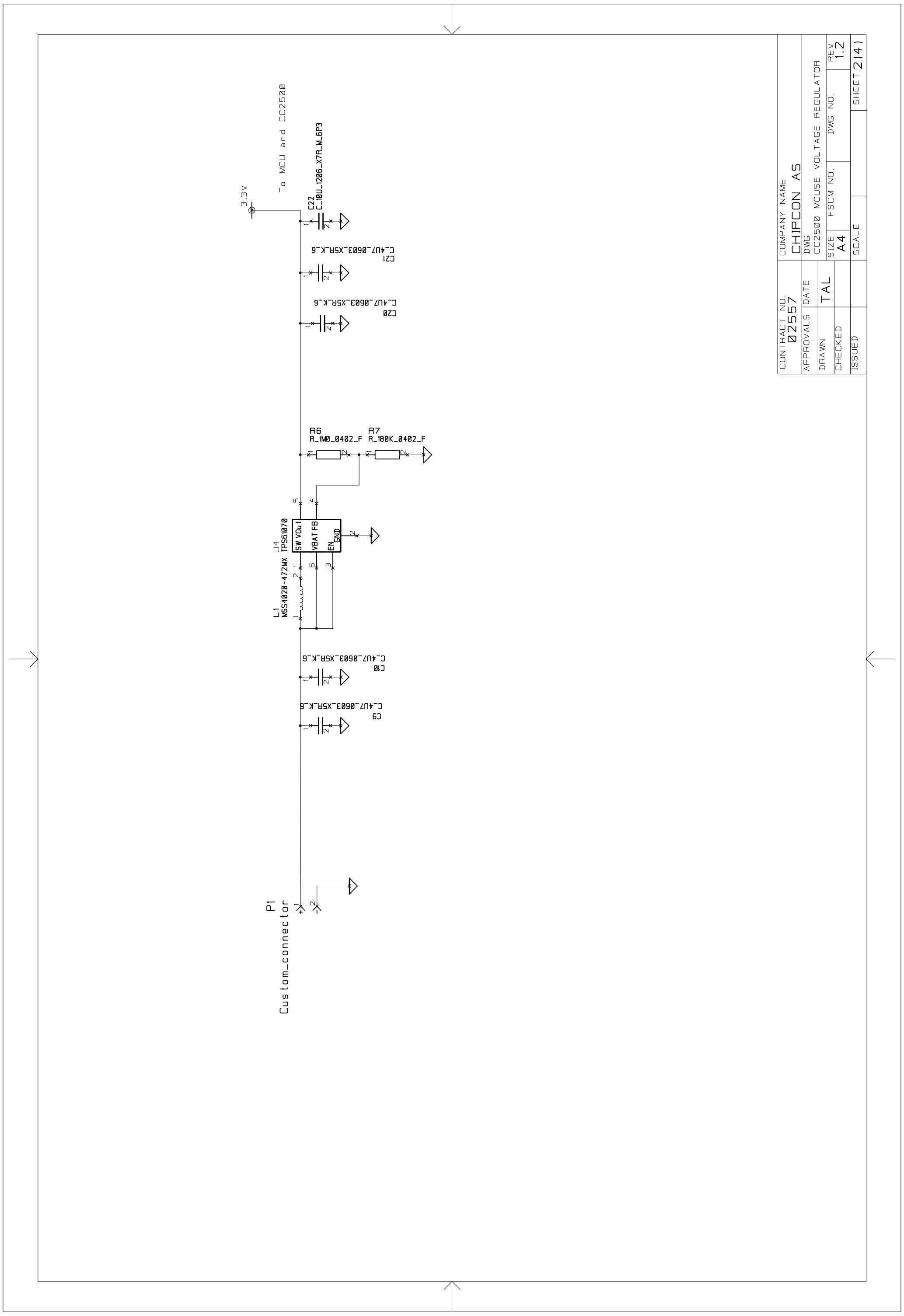


Figure 20. Mouse schematic, page 2

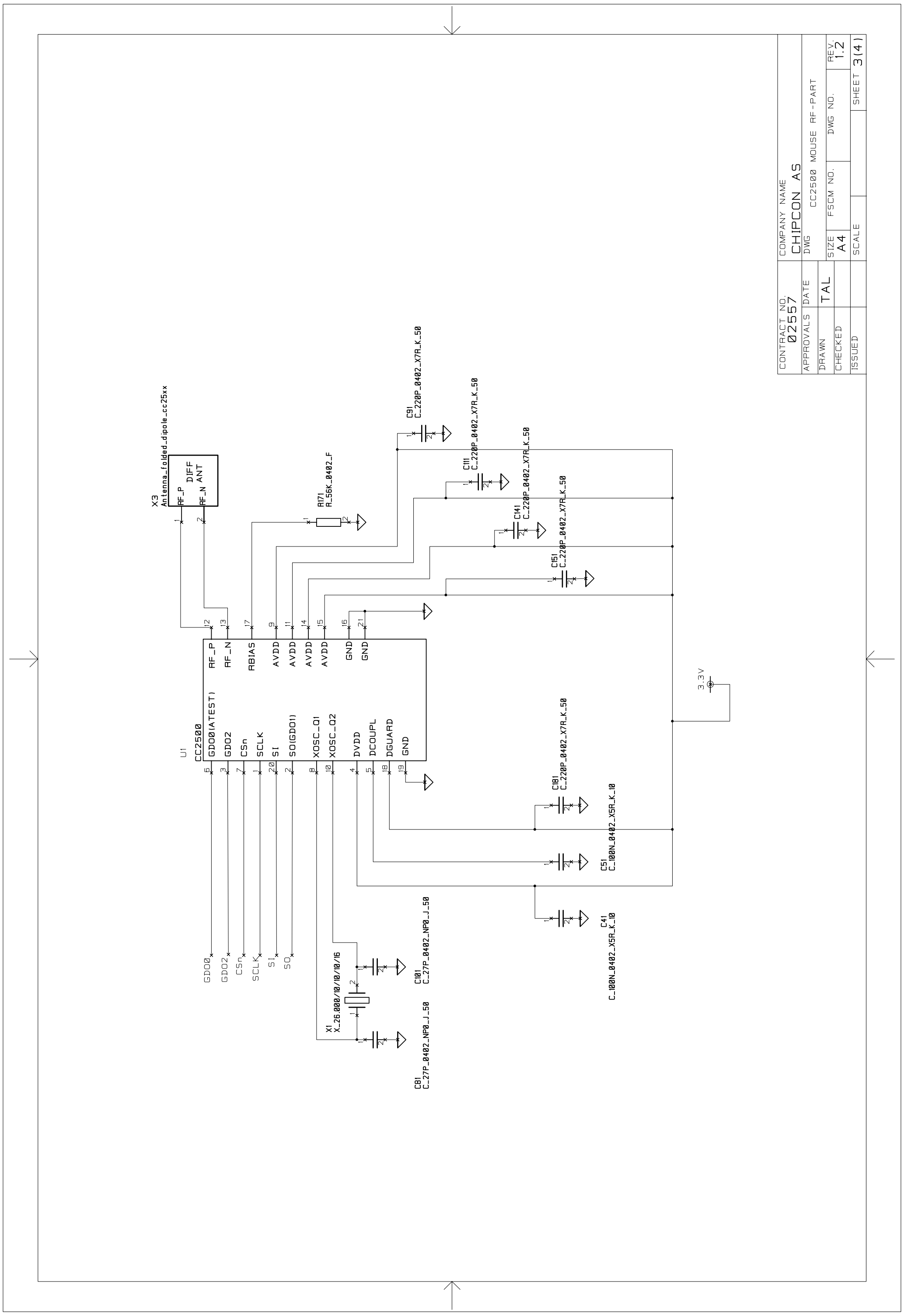


Figure 21. Mouse schematic, page 3

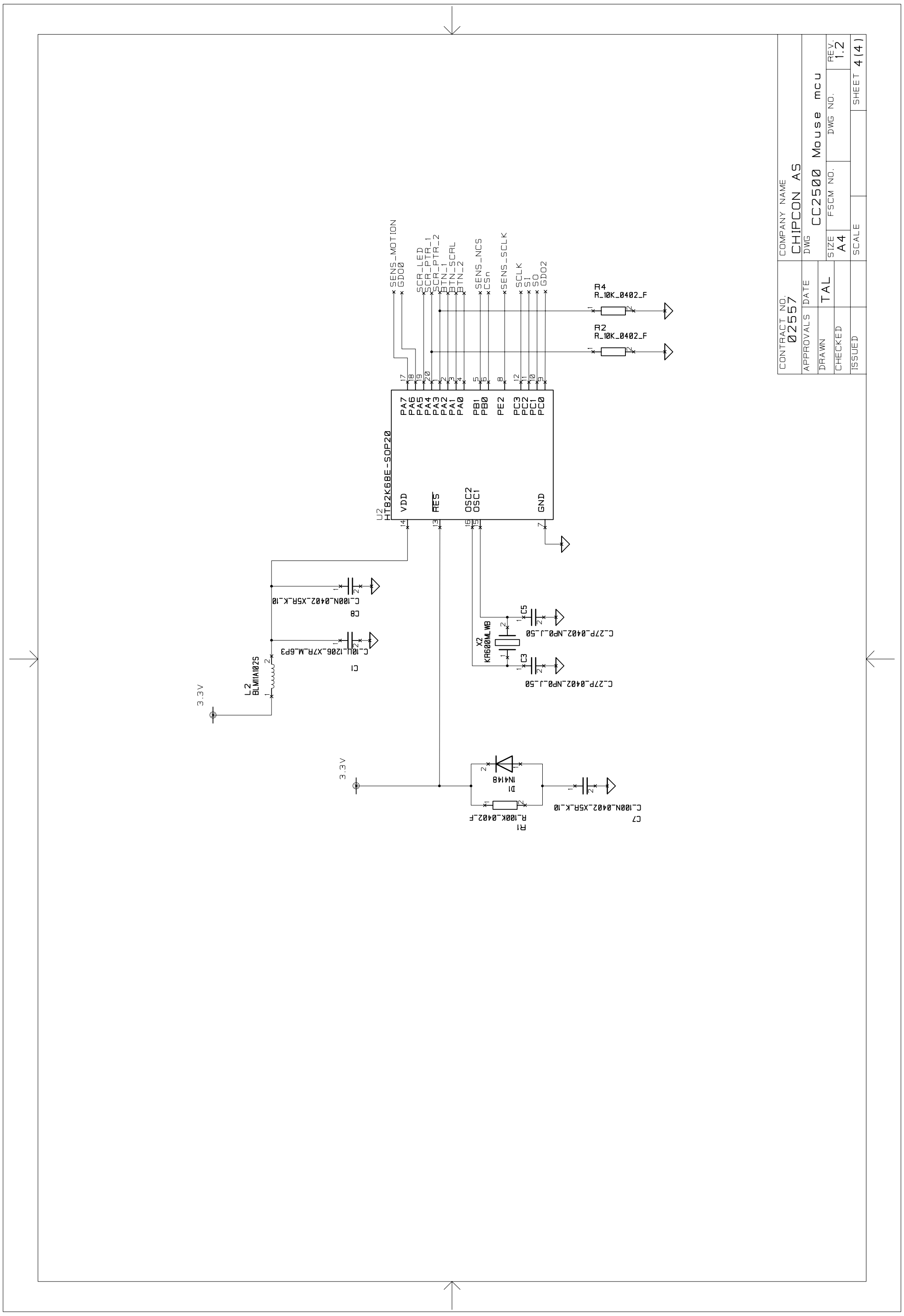


Figure 22. Mouse schematic, page 4

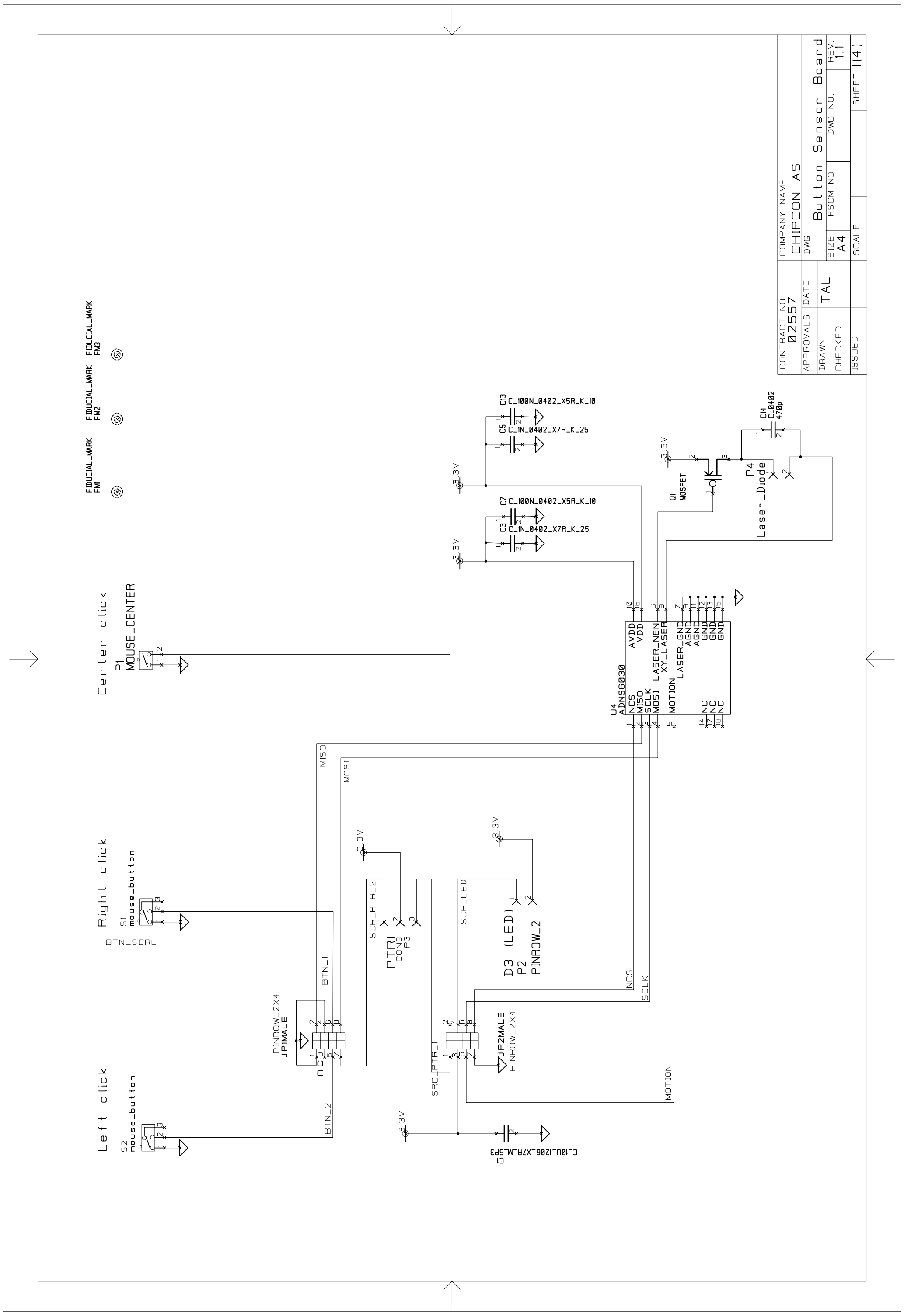


Figure 23. Button and sensor board for laser mouse

Reference	Value/part number	Description	Manufacturer
C1	10 µF, X7R, 20%	Capacitor 1206	
C3	27 pF, NP0, 5%	Capacitor 0402	
C5	27 pF, NP0, 5%	Capacitor 0402	
C7	100 nF, X5R, 10%	Capacitor 0402	
C8	100 nF, X5R, 10%	Capacitor 0402	
C9	4.7 µF, X5R, 10%	Capacitor 0603	
C10	4.7 µF, X5R, 10%	Capacitor 0603	
C20	4.7 µF, X5R, 10%	Capacitor 0603	
C21	4.7 µF, X5R, 10%	Capacitor 0603	
C41	100 nF, X5R, 10%	Capacitor 0402	
C51	100 nF, X5R, 10%	Capacitor 0402	
C81	27 pF, NP0, 5%	Capacitor 0402	
C91	220 pF, X7R, 10%	Capacitor 0402	
C101	27 pF, NP0, 5%	Capacitor 0402	
C111	220 pF, X7R, 10%	Capacitor 0402	
C141	220 pF, X7R, 10%	Capacitor 0402	
C151	220 pF, X7R, 10%	Capacitor 0402	
C181	220 pF, X7R, 10%	Capacitor 0402	
D1	20BQ030	Schottkey diode, 2A	
JP1MALE	PINROW_2X4	Pinrow, 2x4	
JP2MALE	PINROW_2X4	Pinrow, 2x4	
L1	MSS4020-472MX	Power Inductor, 4,7uH	Coilcraft
L2	BLM11A102S	EMI filter bead	Murata
P1		Battery connector	
P2	PINROW_2X5	Pinrow, 2x5	
P3	PINROW_2X5	Pinrow, 2x5	
R1	100 kΩ, 2%	Resistor 0402	
R2	10 kΩ, 2%	Resistor 0402	
R3	82 Ω, 2%	Resistor 0603	
R4	10 kΩ, 2%	Resistor 0402	
R5	10 kΩ, 2%	Resistor 0402	
R6	1 MΩ, 2%	Resistor 0402	
R7	180 kΩ, 2%	Resistor 0402	
R8	10 kΩ, 2%	Resistor 0402	
R171	56 kΩ, 1%	Resistor 0402	
U1	CC2500	2.4 GHz RF transceiver	Chipcon
U2	HT82K68E-SOP20	Keyboard MCU, SOP20	Holtek
U4	TPS61070	Synchronous boost switch	Texas Instruments
X1	X_26.000/20/20/10/16	26.000 MHz Crystal, 20/20 ppm tolerance, HC-49/NX5032-SMD	
X2	KR600MLWB	6MHz resonator	Kingstate

Table 7. Mouse bill of materials

6. The RadioDesk™ reference design firmware

6.1 Overview

The RadioDesk™ reference design firmware runs on the Holtek MCUs used in the mouse, the keyboard and the dongle. The firmware, except the USB-related code and the SPI-interface to the RF chip, is written in C. The firmware is designed with MCU portability in mind. Porting to a different controller would only involve changing the HAL and the aforementioned parts written in assembly.

Note that no software is needed on the PC side; the USB dongle firmware is written to fulfil the USB HID requirements, so the keyboard and mouse will work with any operating system that supports USB Human Interface Devices (HID).

The RadioDesk™ protocol uses adaptive frequency hopping on four active channels picked from 64 possible. The system is always hopping on four channels and a channel is replaced if the packet loss count exceeds a programmed threshold. For instance, a channel will be replaced if jammed by a WLAN system.

6.2 Code structure

The RadioDesk™ firmware is divided into four distinct layers, interfacing each other through a set of functions, macros and definitions. All files, functions and macros have prefixes that clearly indicate which layer they belong to. The exception is the application layer where no prefixes are used.

Layer	Description
HAL	Defines the interface between the MCU and the RF chip, sensors and other IO devices. It also defines the internal registers of the MCU.
RF (PHY)	Defines the properties of the RF communication link. This is typically settings like data rate, modulation formats etc. The RF layer is used by the MAC to establish a link between the master and the slaves.
MAC	Defines how the RadioDesk devices communicate across the RF link. Includes addressing, timing, frequency agility, pairing and retransmission. The RF chip handles some of the MAC functionality; address filtering being an example of this
APP	**Slaves**: Reads data from mouse and keyboard and uses the MAC to communicate across the RF link. **Master:** transfers data between the RF link and the USB interface.

Table 8. Firmware layers

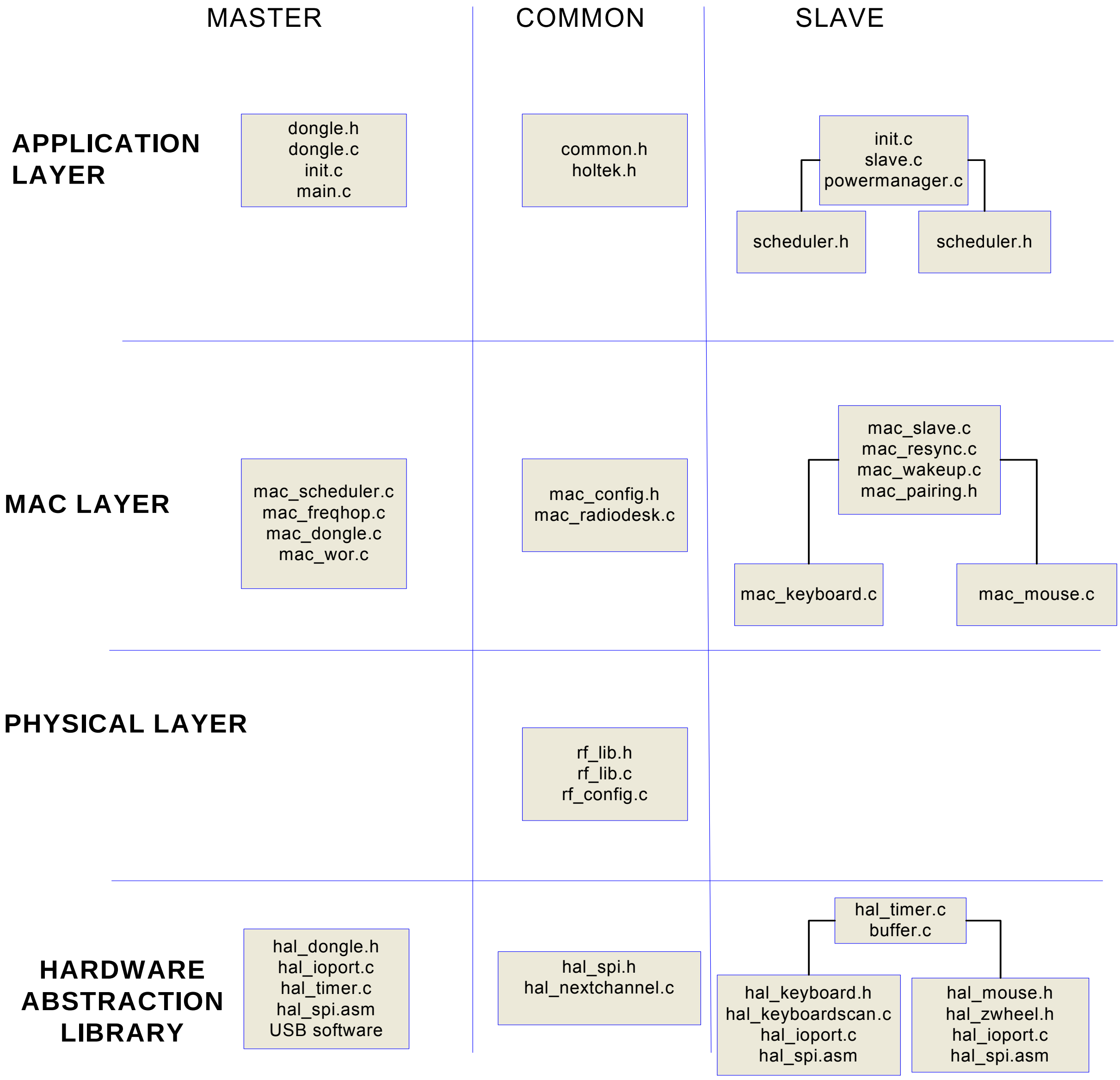


Figure 24. Firmware layers and associated files

6.3 Code generation and maintenance

The RadioDesk™ firmware is delivered as a zip-file. The file contains the code, the documentation and a list of files and functions in the system. This list (inventory.html) contains details on where the files and functions are located as well as a short description of each files contents.

The firmware is built using the HT-IDE3000 development suite from Holtek, which is available from the Holtek WEB site (www.holtek.com). Please contact Holtek for emulator hardware (HT-ICE and interface boards).

It is recommended to unpack the code to the folder **C:\Radiodesk** in order to build directly from the delivery. Otherwise the HT-IDE3000 project files must be reassembled. RadioDesk™ firmware and documentation is delivered together in a ZIP-file.

The directory structure is shown in Figure 25. C:\RadioDesk is the root.

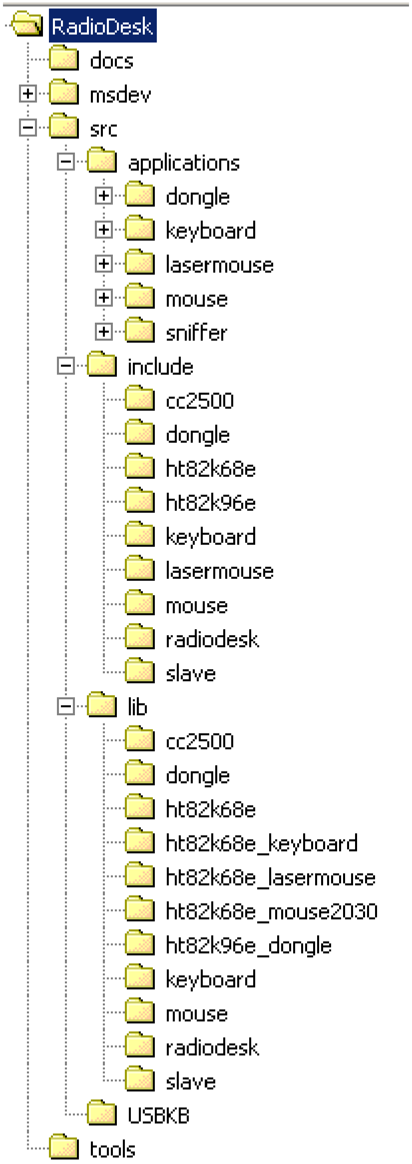


Figure 25. File organisation

The HT-IDE300 project files are located as shown below:

- **./src/applications/dongle/ht82k96e/rd_dongle.prj**
- **./src/applications/keyboard/ht82k68e/rd_keyboard.prj**
- **./src/applications/lasermouse/ht82k68e/rd_lasermouse.prj**

The folder **msdev** contains *Microsoft Visual Studio Project* files for the source code. Note that the code cannot be built using Visual Studio. It is merely there to aid source code navigation., and other code organisers like *Ultra Edit* or *Code Warrior* may equally well be used.

Basic configuration parameters are found in **include/radiodesk/mac_config.h.**

```
//-------------------------------------------------------------------------------
// Network ID (range 0x0000 to 0x7FFF)
#define NETWORK_ID                          0x3008

//-------------------------------------------------------------------------------
// Units to include
#define INCLUDE_KEYBOARD
#define INCLUDE_MOUSE
#define NUM_DEVICES          2
```

The main configurable parameter, NETWORK_ID, must be unique for each dongle in a production batch. The "Units to include" is important for implementers who require a system with only one slave. Uncomment the redundant slave and set NUM_DEVICES to 1.

Basic RF parameters are found in **mac_config.h.**

```
//-------------------------------------------------------------------------------
//  Data rate programming
//#define RF_500KBPS_MSK
#define RF_250KBPS_MSK
//#define FEC_ENABLE

//-------------------------------------------------------------------------------
// Output power (values defined in rf_lib.h)
#define OUTPUT_POWER                OUTPUT_M1_DBM    // -1 DBm
```

RadioDesk™ supports 500-kBit/sec data rate in addition to the default 250 kbps. At 500 kbps the timing allows FEC to be used. The output power is also set in this file.

6.4 MAC functionality

The core of RadioDesk™ is implemented in the MAC layer. Timing properties, device access and frequency agility are all managed here. The MAC is the interface between the applications at both ends of the link; it uses the RF (PHY) layer and the CC2500 to establish a connection over the air.

Timing (synchronisation) is based on the ***beacon*** message, transmitted at the start of each *frame*. The slaves synchronise their clock on detection of the sync word of the beacon. The only other synchronised activity in the slave is the transmission of data back to the master.

Data collection, power management and buffering of data are carried out independently of the synchronised activities.

Device access is based on the network identifier. Every set of RadioDesk units needs a unique ID to allow identification and to avoid interference. The network ID (serial number) is an integral part of the message in both directions. RadioDesk uses address filtering to reject messages from neighbouring RadioDesk networks.

Pairing provides flexibility when combining master and slave units. When powered up, the slaves have no ID and address filtering is disabled. All beacon messages are received but only the first beacon with a sufficiently high signal strength is considered for pairing. If the beacon contents indicate that the master is not paired to any device of this type, the slave will respond and the master subsequently accept the binding request. At this point the slave enables address filtering and beacons from other networks are ignored.

The Frequency agility scheme of RadioDesk is based partly on packet loss, partly on noise measurements. The decision to replace a channel is entirely based on packet loss statistics.

6.5 Master

The USB dongle is the master device of RadioDesk and responsible for synchronisation, frequency selection and data transport. It stays one PTU in each time slot, and the slaves use beacon reception to synchronise with their master. The four states (time slots) are shown in Figure 26. Each state corresponds to one PTU or frame slot. The dongles main responsibility is to transport data from the connected devices to the host computer over USB in a manner, which is transparent to the driver.

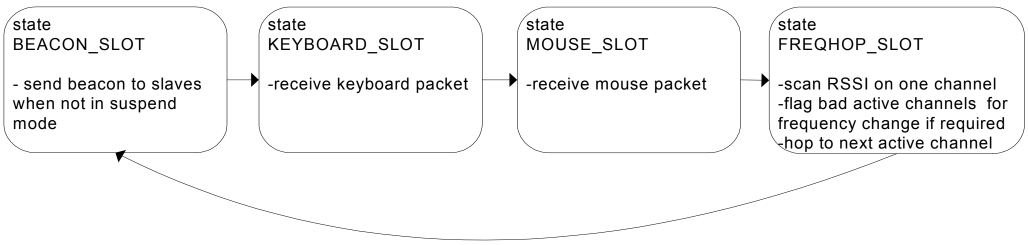


Figure 26. Dongle state diagram

In addition to keeping track of time, the master controls and initiates frequency changes on the active channels. The dongle executes a frequency change if the following criteria are met with respect to the state of the slaves:

1. The dongle has detected packet loss on a channel.
2. All slaves who are in AQUIRE mode have acknowledged the previous frequency change.

Slave modes are shown in Figure 27.

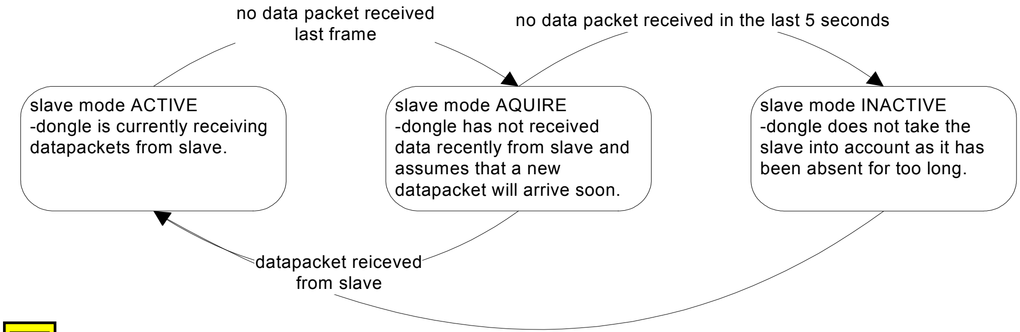


Figure 27. Slave modes

In a normal operating environment, at least one slave will be in AQUIRE mode at any given time. As a result, the dongle has to wait until each slave has been in either ACTIVE mode or INACTIVE mode before executing two frequency changes successively.

Replacement of an active channel is carried out if and only if all of the conditions listed below are satisfied:

- Some channel has reached the packet loss threshold
- No frequency change has taken place after a slave has entered acquire mode
- The RSSI scan has found a free channel
- The new channel is at least 3MHz away from any of the currently active channels
- At least one slave has received the latest beacon

The RSSI scan is carried out every cycle, on the first available channel that is not blocked. Channels are blocked when they are replaced or when there is too much noise detected. The list of blocked channels is periodically cleared.

Figure 28 shows the structure of the dongle firmware. The PC polls the RadioDesk™ dongle for keyboard/mouse data every 10ms (low speed HID device), and the dongle will report whatever data is available.

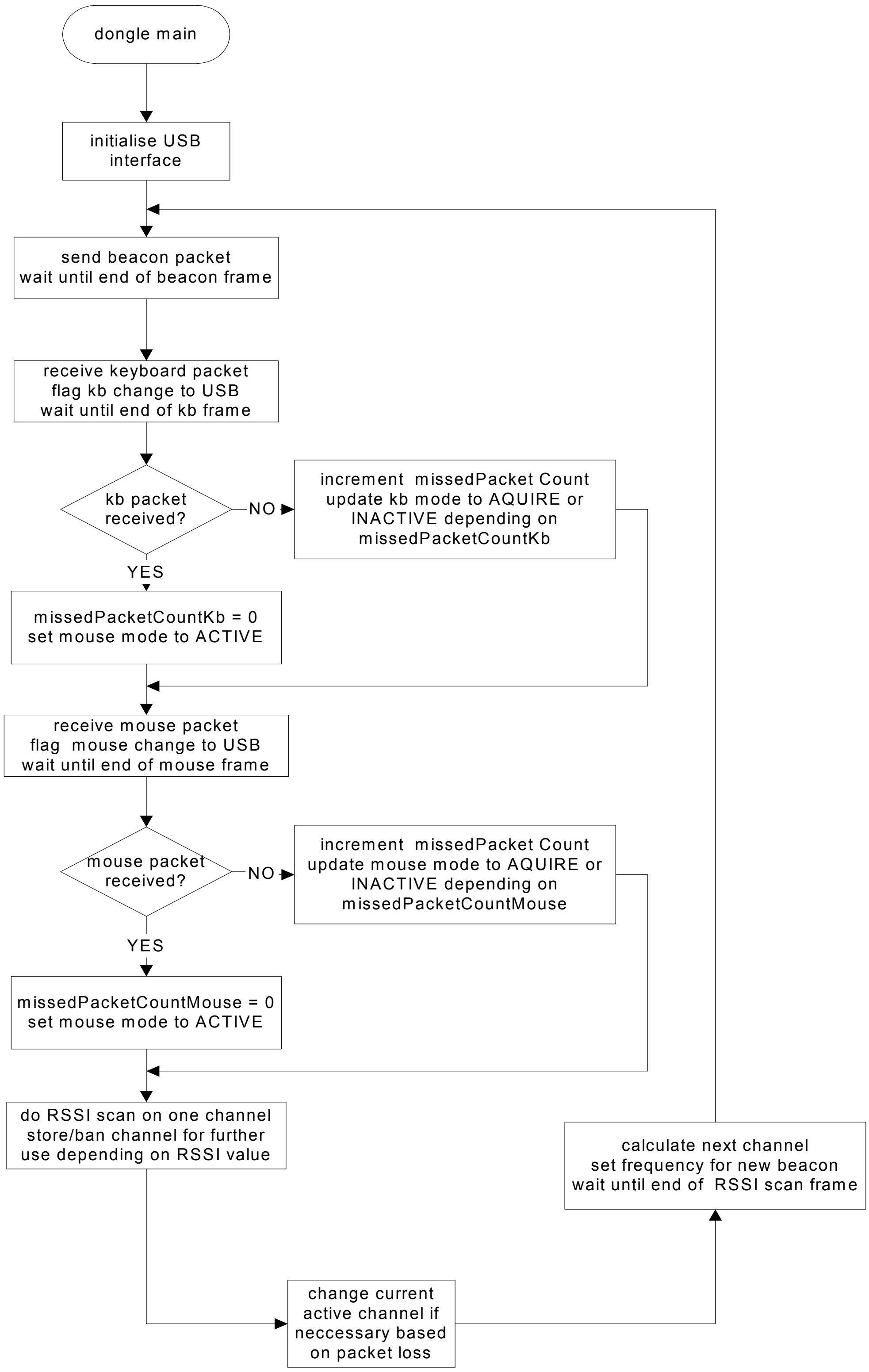


Figure 28. Dongle program flow

6.6 Slave

The slave, when active, listens for beacon packets and transmits its data in the assigned time slot. Apart from packet transmission and reception there are no synchronous activities activity in the slave. The rest of the time, the slave scans inputs (mouse sensor, keys, buttons, etc.) or sleeps to conserve power.

If the slave loses a large number of beacons it tries to resynchronise by scanning the currently active frequencies. If this fails it scans the whole frequency range.

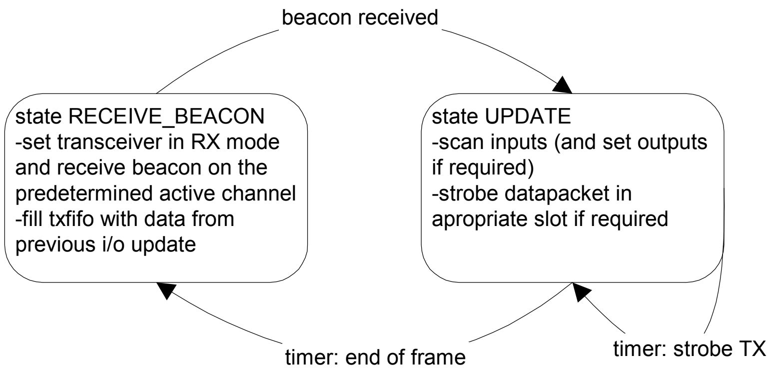


Figure 29. Slave states

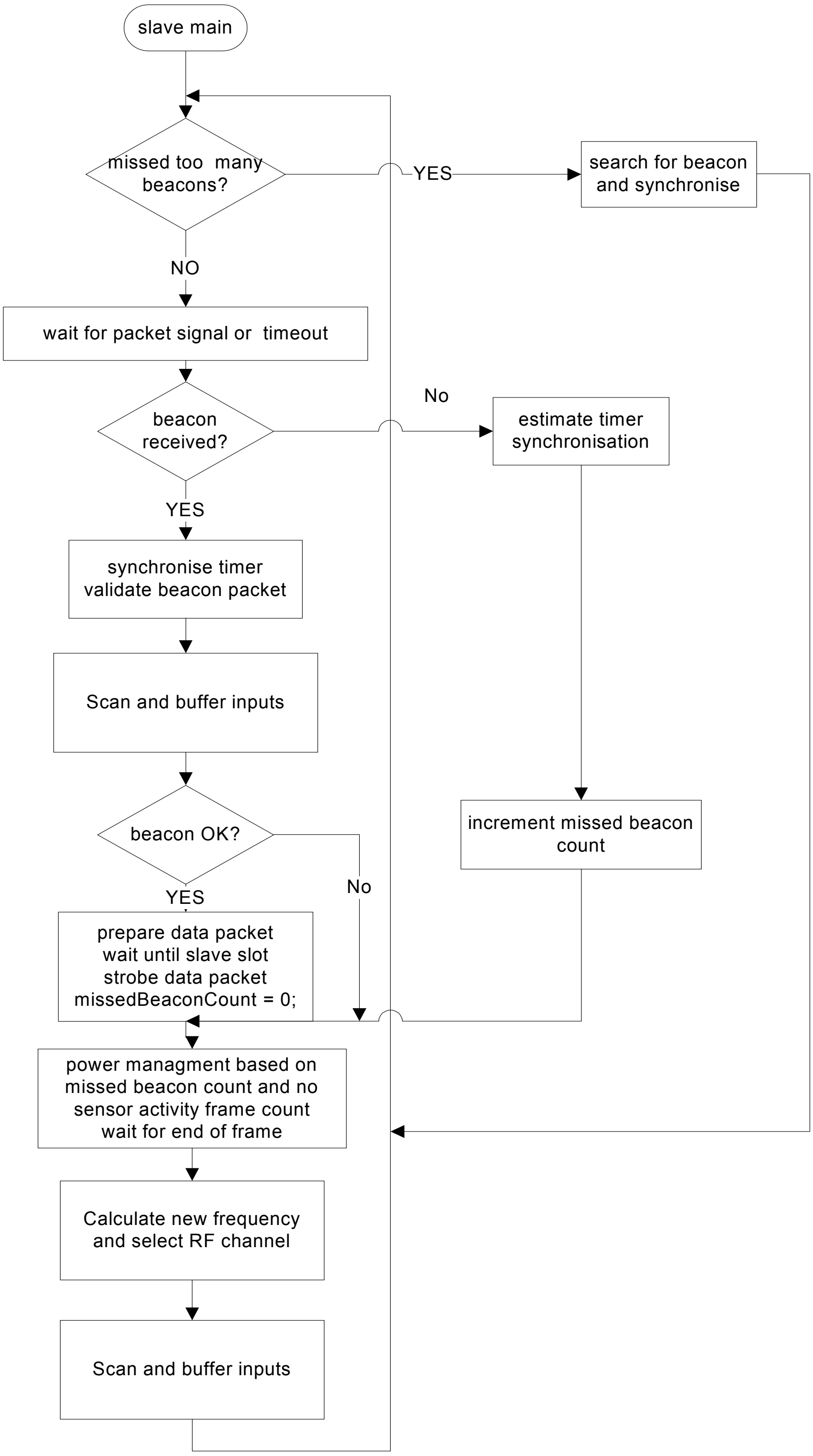


Figure 30. Slave program flow.

6.7 Slave scheduler

Although the RadioDesk™ is delivered with keyboard and mouse scanning firmware, the customer might want to either modify an existing HID device or use RadioDesk™ for other types of devices (game pads, voting buttons etc.).

Customising the slave software requires an understanding of the basic mechanisms of the slave application scheduler. The scheduler tasks that are

1) Executed at a time relative to beacon reception
2) Executed when MCU capacity is available.

Asynchronous tasks include input scanning, buffering and power management. Inputs may in principle be scanned and buffered anywhere in the RadioDesk frame.

The slaves buffer data in order not to lose user input if the link is poor. The keyboard firmware uses a ring buffer for key hits. The mouse code accumulates the movement offsets.

The mouse and keyboard software splits **input scanning** in two parts, one scan just after the power manager and one between beacon reception and data transmission. This gives a sample rate of twice the frame rate. If even higher rates are needed (e.g. scroll wheel or track ball), extra sampling may be used. The scroll wheel requires a sample rate of 100-uSec; hence it is implemented in the interrupt service routine of the tick timer.

The slave scheduler is based on **ticks** of 100-uSec duration, these are synchronised with reception of the beacon. The frame length of 8 ms gives a total of 80 ticks per frame. The beginning and end of the frame are utilised by RadioDesk™ for beacon reception and frequency hopping, the rest is available to the application.

The task names referred to in Table 9 are implemented as macros. The body of the macros executes the task, sets a *complete flag* and then jumps back to the beginning of the main loop Synchronous tasks take a relative tick time as argument.

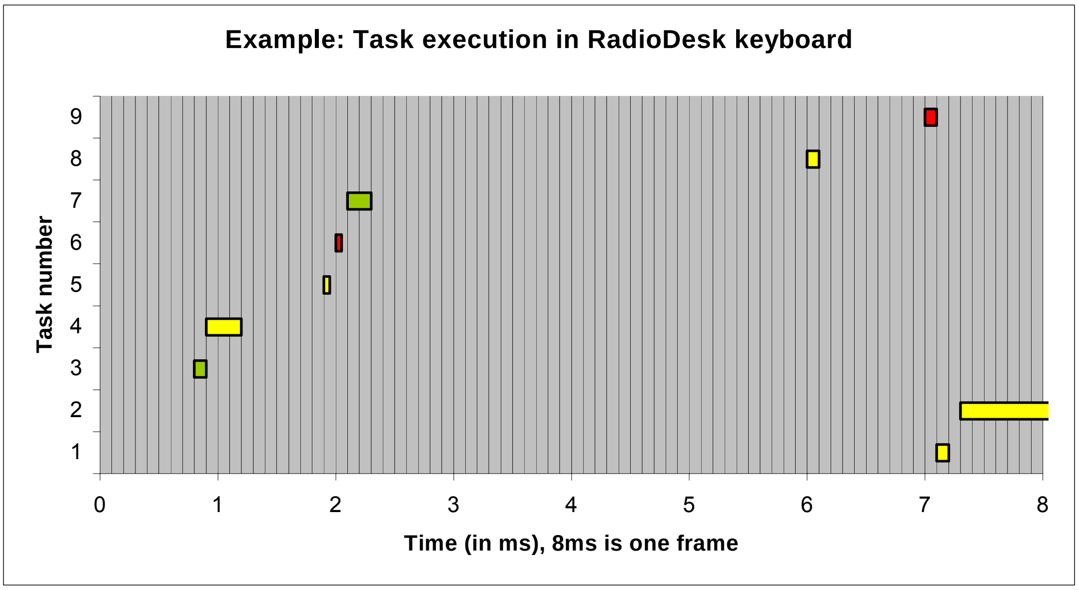
The actual implementation of the scheduler is located in two files: **lib/slave/slave.c** and **include/keyboard/scheduler.h** (or include/mouse/scheduler.h). The task macros are defined in scheduler.h; the calls are done from slave.c.

Table 9 - Slave tasks

Tick	Task name	Description	Mandatory	Synchronous	Type
0	MAC_AQUIRE_BEACON	Listen for, receive and validate beacon	Yes	Yes	SYSTEM
-	DEV_GET_DATA_2	Read and buffer user data	No	No	USER
-	MAC_PUT_DATA	Move data to TX FIFO	Yes	No	SYSTEM
*)	MAC_TRANSMIT_DATA	Transmit data	Yes	Yes	SYSTEM
-	POWER_MANAGER	Power management	No	No	SYSTEM
-	DEV_GET_DATA_1	Read and buffer user data	Yes	No	USER
65	MAC_SWITCH_CHANNEL	Frequency change.	Yes	Yes	SYSTEM
70	MAC_START_RX	Prepare for beacon reception	Yes	Yes	SYSTEM
75	MAC_PUT_HEADER	Move packet header to TX FIFO	Yes	No	SYSTEM

*) Tick depends of the time slot for the slave: 19 for keyboard, 39 for mouse.

Task name	Task	Tstart	Tstop	Duration	T between tasks	Comment
MAC_PUT_HEADER	1	7,1	7,2	0,1	0,1	
MAC_AQUIRE_BEACON	2	7,3	0,8	1,5	0	
DEV_GET_DATA_2	3	0,8	0,9	0,1	0	
MAC_PUT_DATA	4	0,9	1,2	0,3	0,7	
MAC_TRANSMIT_DATA	5	1,9	1,9	0,05	0,1	duration estimated
POWER_MANAGER	6	2	2	0,05	0,1	duration estimated
DEV_GET_DATA_1	7	2,1	2,3	0,2	3,7	
MAC_SWITCH_CHANNEL	8	6	6,1	0,1	0,9	
MAC_START_RX	9	7	7,1	0,1		
SCHEDULER_RESTART	10					
			Total:	2,5	5,6	



Important notes:
1) Timing reference (T=0) is when beacon data receiving is started (after sync word has been detected)
2) The measurements above are done by examining the timer value of the slave (keyboard). This value is incremented each 100uS (resolution is therefore 100uS), and there are 80 timer ticks in one frame. Hence, tasks which execute in less time than 100uS might appear to take no time (which is the case for MAC_TRANSMIT_DATA, for instance).
3) This is actual execution time on the MCU and does not take into account neccessary idle time for radio and i/o devices to complete their tasks (beacon sending etc).
4) As we can see, only 2,4/8 = 30% of the CPU time is used on synchronous and asynchronous tasks in the keyboard. **Code executed in the timer interrupt is added to this!!**

Color codes:

Green	Asynchronous, not mandatory task
Yellow	Asynchronous, mandatory task
Red	Synchronous, mandatory task

6.8 Adding slave features

The keyboard and mouse firmware delivered with RadioDesk™ contains the basic functionality of standard keyboard and mice. The keyboard implements standard keys only. Extensions like media keys, scroll wheels and trackballs are not included as these may vary between implementations. The mouse implements movement, three standard buttons and the scroll wheel.

Example: Adding trackball code to a keyboard

To make this example simple we assume that the last two bytes of the keyboard packet can be used to transfer trackball movement data to the master. The framework for adding trackball code is in place, the file **mac_config.h** contains a definition, which controls whether a keyboard has trackball functionality or not. The definition is commented out by default. Furthermore, the implementer must add this own trackball sampling code in the interrupt service routine for the timer. This may be done in the macro *IO_POLL()* **hal_keyboard.h**.

6.9 Power management

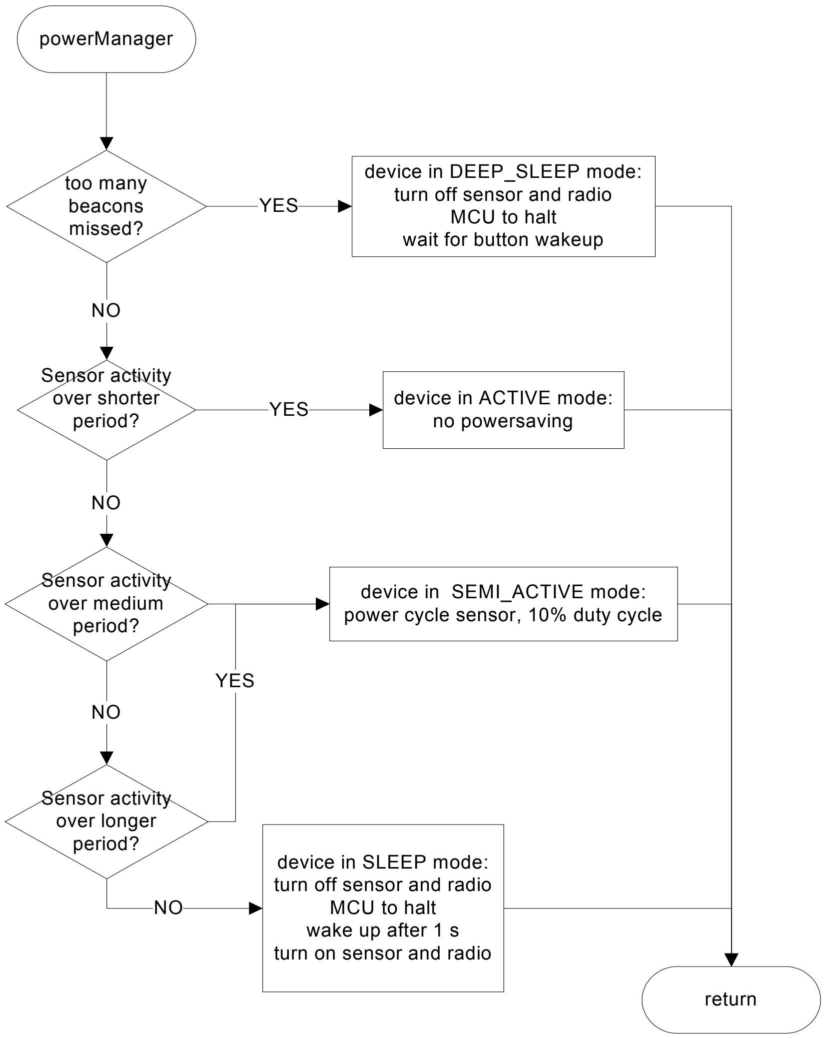


Figure 31. Power management

The parameters for power management are configurable from device to device as the usage patterns of a keyboard and a mouse differ. The keyboard skips the semi-active mode entirely and enters sleep after a given timeout. Configuration is found in the file *scheduler.h* for the device in question.

7. Modifications

7.1 Suggestions to lower price

- The USB dongle could be changed to a “hockey puck” design to save cost of balun circuit and chip antenna.
- Three AAA cells could be used in the mouse instead of two AA cells. This will shorten the battery lifetime, but make it possible to use a linear regulator instead of the switch-mode regulator used in the standard design.

7.2 Suggestions for longer battery life time

- Require the user to wake up the mouse by pressing a key when the mouse has been inactive for a long time. This allows you to shut off both the radio and the optical sensor, increasing the battery lifetime.
- Use more than two AA cells in the keyboard to extend the battery lifetime.
- Increasing the RF data rate to 500 kbps will reduce the on time of the radio, reducing the current consumption somewhat.
- reduce radio output power

7.3 Suggestions for advanced functionality

- USB dongle could be modified to add PS/2 option.
- To take full advantage of the two-way radio link, it is possible to include an LCD display on the keyboard. This can indicate caps lock, num lock and scroll lock status, as well as time and date, currently playing artist and song for a Media PC, calculator etc.
- Force-feedback technology can be added to the mouse in order to alert the user about system warnings etc. This would use standard force-feedback actuators such as those used in game pads.
- Wireless desktop (wireless keyboard and mouse) could add a third remote control unit, especially useful for Media PCs. This would entail adding a time slot to the frame format.
- It is possible to extend the range of the RF link by increasing the RF output power. This can be done by adding two T/R switches and a single-transistor power amplifier to the RF circuitry. This will require the use of single-ended antennas, and it is important to consider regulatory requirements as well.
- Another possibility for extending the range is to use a lower RF data rate. This will increase the latency and the power consumption, but will allow a longer RF range without increasing the output power.
- A third possibility is to utilize a different RF frequency. For instance, it is quite simply to port the existing design to the CC1100 to work in the US 902-928 MHz frequency band, for instance. The porting would involve changing the RF frequency settings and re-routing the PCBs, since antennas and matching circuitry are different at 900 MHz.

7.4 Suggestions for applying RadioDesk™ to other applications

- Gaming controllers (joysticks, gamepads, dance mats, motion controllers etc)
- Remote controls
- Other star networks where master is not power-limited

8. Document history

Revision	Date	Description/Changes
1.0	2005-05-04	Initial release
1.1	2005-07-18	Update for firmware rev. 0.97
1.2	2005-07-27	Update for firmware rev 0.98
1.3	2005-09-16	Update for firmware rev. 1.02