DINAMO Model Railroad Control

MCCdec

Manual MCCdec 02

Author: Leon J.A. van Perlo
Version: 2.0
Date: November 16th, 2008

Release management

This manual is applicable for modules and kits consisting of:

- Print
  - MCCdec Rev02
- Firmware
  - MCCdec Rel 2.0

This document is written using OpenOffice.org Writer and generated with CutePDF

Contents

1 Introduction

On a model rail road layout nowadays one finds more often an extension with moving model-cars, for example on the basis of the Faller Car System. The possibilities to control cars of this system in the original state are very limited. Cars are stopped by means of a magnetic field generated by a coil underneath the road surface, actuating a reed-contact that interrupts the current to the motor, forcing a sudden stop. Other control-possibilities are almost completely absent.

On behalf of Railz Miniworld in Rotterdam, a method has been developed to control these kind of cars in a sophisticated way. Cars are equipped with a decoder that receives commands through a wireless communication system. The signals and commands for the decoders can be generated by the Dinamo control system.

The control-possibilities of such cars include:

- Speed control in 15 steps
- Cruise control (constant speed)
- Automatic slow acceleration and deceleration (mass-simulation) with different characteristics
- Control of headlights, brakes, tail lights, blinkers and additional functions.

Further the decoder has additional features, such as:

- Emergency stop in case the car gets off-road
- Battery-monitoring
- Sleep-mode
- Extensive configuration possibilities

The decoder can be configured for each car to meet the desired characteristics of that specific car. Configuration is achieved by means of 'Configuration Variables', a term borrowed from the DCC system. Configuration Variables are values, stored in a non-volatile part of the decoders memory, by which the decoder is given a specific behaviour.

2 Hardware and Software versions

At this moment, the MCCdec exists in different revisions and releases.
The revision level refers to the type of PCB, so the hardware. The version-level refers to the software.

2.1 Hardware

MCCdec-rev00 is the first 'commercially' available production.
Dimensions: 25,5 x 16,7 x 2,8 mm
This PCB is a single sided assembly.
MCCdec-rev00 is no longer in production

MCCdec-rev01 is electrically 99% identical to rev00. The main difference are the dimensions.
Dimensions: 17,3 x 12,8 x 3,9 mm
This PCB is a double sided assembly.
2 components that have to be added externally with rev00 are included in the rev01 PCB.

MCCdec-rev02 is electrically and functionally substantially different from previous versions. While rev00 and rev00 work directly on the battery power, rev02 has a power converter by which the decoder internally runs on a fixed voltage of 4.1V. This has two advantages: The decoder can run on battery power as low as 0,9V (minimum start voltage = 1,0V). Second all outputs have a 4V capability allowing every output to drive white and blue LEDs directly
Dimensions: 17,3 x 12,8 x 4,3 mm

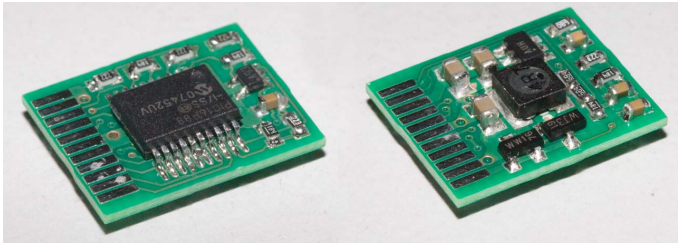

Fig 1: MCCdec rev02

2.2 Software

MCCdec-rel 1.0
The first 'commercially' available version.

MCCdec-rel 1.1
Has an improved motor-control and a special feature to control motors with higher induction.

MCCdec-rel 1.11
Improved battery-management.
Further the factory-settings of this release are optimized for a battery voltage of 2.4V

MCCdec-rel 2.0
Fully revised software for MCCdec-rev02

2.3 Compatibility

Software rel 1.x runs on decoder rev00 and rev01
Software rel 2.0 runs on decoder rev02

MCCdec-rev02 with software rel 2.0 is referred to as MCCdec02

Note: This manual only covers MCCdec-rev02 and software rel 2.0. For other versions, please use previous versions of this manual

3 Communication

3.1 Introduction

The MCCdec decoder receives instructions from a central control system via a wireless connection. Transfer of data is by means of induction. The signal is picked-up by the decoder by a small coil, fitted underneath or in the base of the car.

Information-transfer is by means of addressable packets in a continuous data stream. As soon as a command for a car is modified, the corresponding packet is inserted with priority in the data stream. All packets for all active cars are cyclically repeated so every car can pick up the required status any time, e.g. after packet-loss.
Packets include an integrity-check to verify the received data is correct and to prevent mutilated packets to have undesired effects. Packets with false integrity are ignored completely. Every decoder is configured with a unique address by which the car can filter 'his' data from the data stream.

Besides the cyclic regular packets, there are so called one-off packets (transmitted only once), “Idle” packets (that are recognized as valid data stream, but do not contain any information) and broadcast-packets with information intended for every receiver.

3.2 Address and type

Every packet contains data regarding address and type. The address indicates for which decoder the packet is intended. The type indicates the basic function of the packet.

The address consists of 12 bits. The address range is 1..4095. Address 0 is interpreted as 'broadcast', meaning: intended for all decoders.
The type consists of 4 bits (range 0..15). At the moment the following types have been defined:

- 0: Normal operational packet
- 1: Configuration-start
- 2: Configuration-data
- 3: Configuration-end
- 15: “Idle” (no function)

3.3 Data (operational)

Normal packets (packets that are sent to the decoders during normal traffic operation to control the cars) contain the following information:

Speed and acceleration:

- Speed (0..15)
- Direction (bit), forward=1, reverse=0, not used at this moment
- Acceleration-selector (0..7) for smooth acceleration/deceleration

Functions:

- H (bit): Headlights
- B (bit): Brake[1]
- L (bit): blink Left
- R (bit): blink Right

[1] As an alternative the car can control the brakes autonomously while slowing-down. In that case this function can be used to control the brakes while the car is stopped.

- F4..F1 (bit): 4 Additional functions, such as gyrolights, interior lighting, sound, etc, (depending on the features of the decoder or additional hardware) or controlling the headlights separately.

3.4 Configuration

To configure the decoder, 3 types of configuration-packets are defined. Decoder-configuration happens in page-mode. The data for multiple Configuration Variables are loaded in the decoder and after verification written in Flash memory in a single action.

CV's are split in 4 'pages':

Page 0 = CV 0..63
Page 1 = CV 64..127
Page 2 = CV128..191
Page 3 = CV 192..255

Configuration requires a sequence of packets to be received by the decoder in the right order and without interruption by other packets[2]:

Configuration-start

The configuration-start packet puts the decoder in configuration-mode. It stops the vehicle and switches-off all functions (if not already done).
The configuration-start packet contains which CV-page shall be written and how many variables will follow.

Configuration-data

The configuration-data packet contains the CV number (0..63) within the active page and the value (0..255) to be written in the variable.
The received data is not saved in the CV yet , but buffered in RAM until reception of the configuration-finish packet.

Configuration-finish

The configuration-finish packet contains checksum-data to verify that the sequence start-data-finish has passed correctly and without errors. If that is the case, the received data are actually stored in Flash-memory of the decoder.
During this write-cycle the blinkers (left and right) are switched on. Hereby the blinkers give a short flash as a visual indication that the configuration sequence was successful.

Any error in the sequence aborts the configuration procedure. As a consequence all configuration-data of the current configuration session is discarded.

[2] To be clear: Packets addressed to other decoders may be sent in between as well as 'Idle' packets, but no other packets for the decoder under configuration and no broadcasts.

4 Configuration Variables

4.1 Factory Reset

Writing CV 0 has a slightly different effect. Writing CV 0 puts all CV's in factory settings. The value written in CV 0 has at this moment no meaning. The factory-reset takes place if CV 0 is included in the range of CV's in the configuration-sequence. In that case the factory reset occurs before any other CV is written, so you can reset-and-rewrite in one single sequence if desired.

4.2 Address

Every MCCdec needs to have a unique address within the range of the layout it is operated on. An address is 1...4095.(12 bits)

CV 2 = address-low (8 bits)
CV 3 = address-high (4 bits)

Address = 256 *CV3 + CV2

4.3 Speed Table

The MCCdec has 16 speed steps for the vehicle. Step 0 is speed 0, which leaves 15 steps for other speeds. Every step can have a speed-setting between 0 and 255.

Every speed-setting corresponds to an EMF-value. The EMF is the Voltage the motor generates itself as a result of the rotation of the motor. The higher the rotation speed, the higher the EMF. The speed-setting therefore actually indicates the EMF value the PID controller wants to measure (see further) at that speed-setting.

To achieve reasonable accuracy the speed versus EMF settings are different for 'normal' and 'low' voltage operation. When the minimum battery voltage is configuered to be more than 1,5V (setting 92 or higher), the following table indicates which EMF value corresponds to which speed-setting. In practice therefore the table is valid when using 2 cells or more. Intermediate values exist as well and can be found by interpolating the values given in the table below.

Speed	EMF voltage (V)	Speed	EMF voltage (V)
252	4,1	120	2,0
240	4,0	108	1,8
228	3,8	96	1,6
216	3,6	84	1,4
204	3,4	72	1,2
192	3,2	60	1,0
180	3,0	48	0,8
168	2,8	36	0,6
156	2,6	24	0,4
144	2,4	12	0,2
132	2,2	0	0,0

Table 1: Speed setting versus EMF voltage at minimum battery voltage over 1,5V

Attention: The maximum usable EMF is determined by the actual battery voltage. If e.g. the battery voltage is 2.4V, the maximum usable speed-setting is approx. 140. If the battery voltage drops because it runs empty, the speed-setting 140 will no longer be reached.

When the minimum battery voltage is configured to be less than 1,5V (setting 91 or lower), the following table is applicable:

Speed	EMF voltage (V)	Speed	EMF voltage (V)
252	2,1	120	1,0
240	2,0	108	0,9
228	1,9	96	0,8
216	1,8	84	0,7
204	1,7	72	0,6
192	1,6	60	0,5
180	1,5	48	0,4
168	1,4	36	0,3
156	1,3	24	0,2
144	1,2	12	0,1
132	1,1	0	0,0

Table 2: Speed setting versus EMF voltage at minimum battery voltage below 1,5V

Of course the EMF says very little about the actual speed of the vehicle. This depends very much on the characteristics of the motor, the transmission, wheel size, to mention a few factors. A car with a 2V engine at speed '100' thus can actually go faster than a car with a 3V engine at 'speed' 120. Therefore: try, test and measure the speed yourself.

Step 0 MUST always be 0 (the decoder will not accept another value for this setpoint). The other 15 steps can be filled with setpoints as desired, however it makes some sense to give a higher step also a higher speed-setpoint. It does not have to be linear though, e.g. of you want fine speed-control at lower speeds you can choose to do so.

CV 16 = Speed step 0
CV 17 = Speed step 1
…
CV 31 = Speed step 15

4.4 Acceleration / deceleration

At speed change the vehicle can accelerate/decelerate autonomously. There are 8 acceleration-characteristics to which you can assign an acceleration/deceleration delay (mass-simulation). The higher the value, the slower the change will be.
In principle selector 0 equals 0 (no delay, select new speed directly) and a higher selector a slower change. However this is no rule, you can choose what you want.
How large the delay actually is depends also on the properties of the motor and the speed-table. As a general rule-of-thumb: FIRST tune the speed-table according to your needs, THEN set the acceleration values.

Attention: While accelerating/decelerating also the intermediate speed-setpoints are used. So accelerating from step 3 = 15 to step 4 = 20 makes the decoder to use the intermediate setpoints 16, 17, 18 en 19 to reach the best possible 'smooth' behaviour.

Bit 7 of each acceleration variable (= +128) determines whether during slow-down the brake-lights shall be activated. If this bit is set, the brakes are active as long as the vehicle is slowing down using this parameter. The idea behind this is that with slow changes (less

throttle or no throttle) the brake lights stay off, while with heavier deceleration (real braking) the lights come on.
The above only works when the Auto-brake option is set (CV 45). In that case during movement (speed <> 0) the brake-light is controlled by bit 7 of the delay value. When the car is stopped the brake-light is controlled by the brake function bit.

CV 8 = Selector 0
CV 9 = Selector 1
…
CV15 = Selector 7

4.5 PID control

The PID (Proportional, Integral, Derivative) controller continuously operates to keep the actual speed equal to the desired speed as much as possible. The PID controller is configurable with 4 parameters. Depending on e.g. motor type and weight of the car these parameters can be adapted to get the most optimal behaviour.

The P-factor forms the basic control of the regulator. It is the gain by which the difference between actual speed and setpoint is used to make corrections. A too low P-factor can lead to not reaching the desired speed fast enough, or not at all. A too high P-factor can lead to oscillation.

The I-factor eliminates remaining errors and is important for load regulation. Mathematically the I factor integrates the measured error and uses the result to control the motor. The I-regulator is by nature much slower than the P regulator. A too low I-factor leads to the behavior that speed is not corrected quick enough when the load is changed (e.g. going up-hill). A too high I-factor can result in instability (oscillation).

The D-factor is a fine-tuning parameter and influences the dynamics of the controller. If the car does not run smoothly and you can't correct it by the P and I parameters you can try to improve behaviour with the D parameter.

Finally there is a PID-Control byte of which 6 bits have a function.

- Bit 0 of the PID Control Byte activates the EMF input filter. Some motors don't generate a stable EMF. The input filter can eliminate noise on the EMF signal. The drawback is that control becomes less direct. The input filter has a substantial effect on the control, so when you switch the input filter on or off you'll probably have to modify the PID values as well.
- Bit 1 of the PID Control Byte activates the EMF output filter. The output of the PID controller herewith passes a low-passfilter making the motor control slightly less direct, which decreases motor-hum. The output filter only has a minor effect on the control parameters compared to the input filter.
- Bit 4..7 represent the EMF-timeout (0..15). The motor is pulse-width modulated at high frequency. The EMF measurement happens in between 2 pulses. With most motors this works fine, however some motors have such high induction that the measurement time is too short. This time can be extended by the EMF timeout.
  A too short EMF timeout is recognized by extreme instability that cannot be corrected in any way. If this is the case, put the EMF-timeout at a rather high value (8 or so). Tune the PID parameters and if it works as desired, decrease the EMF timeout to the lowest possible value on which the car behaves smoothly.
  A too high EMF-timeout results in somewhat more motor-hum and possibly somewhat more wear.

CV 32 = PID control (0..243, default 2)
CV 33 = PID P-factor (0..63, default 16)
CV 34 = PID I –factor (0..7, default 4)
CV 35 = PID D-factor (0..63, default 8)

4.6 Battery Monitor

The MCCdec has a precision reference voltage against which the battery voltage is measured. If the battery voltage drops below a minimum setpoint the car can switch to an emergency program. This means that the maximum speed is decreased and alarmlights (both blinkers) are switched on. The (substantially) lowered maximum speed could be recognized by the control software (PC) on which action could be undertaken, e.g. by directing the car to a refill-station

The minimum voltage can be set by CV5. The relation between the voltage and the parameter is found in table 3 below. Intermediate values work as well, you'll have to interpolate yourself.
Note that the minimum battery setpoint influences the speed-EMF table. The change-over is at setting 92 (see par 4.3 for details).

An NiMH cell has a nominal voltage of around 1,25V. If the cell is nearly empty the voltage drops rapidly. The absolute low-value is approximately 0,9V. If you keep drawing current below that level, the cell may get damaged. It makes sense to set the alarm level at 1,1V or 1,15V per cell.

If the battery is “empty”, the maximum speed will be limited to the speedstep given in CV6. If the car goes faster at the moment that event occurs, it will slow down according the acceleration selector also given in CV6. Bit 4 of this CV selects the alarm lights (blinkers) during battery-low status.

If you don't want to use battery-monitoring, set CV5 to 0 when using one cell or 92 when using 2 cells or more.

Setpoint	Battery voltage limit (V)	Setpoint	Battery voltage limit (V)
252	4,1	120	2,0
240	4,0	108	1,8
228	3,8	96	1,6
216	3,6	84	1,4
204	3,4	72	1,2
192	3,2	60	1,0
180	3,0	48	0,8
168	2,8	36	0,6
156	2,6	24	0,4
144	2,4	12	0,2
132	2,2	0	0,0

Table 3: Values for setting the minimum battery Voltage

CV 5 = Minimum battery voltage (0..255, default 64 = 1,05V)
CV 6 = V.V.V.A.S.S.S.S (default 150: V.V.V = 4, A = 1, S.S.S.S = 6)
S.S.S.S : 4 bits giving the maximum speed during battery-low status
V.V.V : The deceleration if the car needs to brake when the alarm status occurs
A : 1 = Alarm-lights on, 0 = alarm-lights off
The decimal value of CV6 = speed + (16 * alarm-lights) + (32 * deceleration)

4.7 Time-out

If the MCCdec receives no signal while the car moves, the car will perform an emergency stop after a certain amount of time. The car is then immediately stopped, all functions are switched-off and the alarm-lights start blinking. The emergency-stop is meant to prevent the car from ongoing uncontrolled movement after signal loss.
The timeout is speed dependent. The lower the speed, the the longer the timeout. This results more or less in a constant distance the car can travel after signal-loss. The timeout is configurable with a parameter between 0 (timeout switched off) and 255. As an indication: at a timeout of 128 the time is approximately 2 seconds at speed 120.
The timeout restarts every time the decoder receives a valid and correctly addressed packet.

CV 7 = Timeout (0..255)

4.8 Outputs and Functions

MCCdec02 has a number of additional functions that can be controlled by function bits in the communication protocol and by 'internal' events. The actual behaviour can be configured.

4.8.1 Outputs

Available outputs:
- X0, X1, X2, X3, X4, X5, X6
- Y0, Y1

So in total 9 outputs are available.

Outputs Y are can only be used for blinkers.

The X outputs can be more or less freely configured.
- X0 through X5 can have 4 states: off, on, modulated(12,5% on) and blink
- X2 through X5 can be controlled by a sequencer.
- X6 has an on/off function only

X6 is combined with the receive indicator function. If X6 is not configured it functions as receive indicator. If X6 is configured by any other function, the receive indicator function is disabled.

By default the outputs are configured to have the following functions:
- X0 = headlights
- X1 = combined brake/tail lights
- X2 = tail/contour lights
- X3 = gyrolight
- X4 = gyrolight
- X5 = additional brake light
- X6 = receive indicator
- Y0 = left blinker
- Y1 = right blinker

Priorities are:
1) when sequencer is activated all other functions are disabled for that output
2) blink overrides on
3) on overrides modulated
4) modulated overrides blink-dark period

4.8.2 Sequencer

The sequencer consists of 16 steps which are executed cyclically. The total timing is configurable. Every sequencer step defines the state of 4 outputs X2..X5, that is, when the sequencer function is activated for these outputs. The total sequencer is described by 17 CV's.:
CV 39 = sequencer time.
Total sequence time = 80 * (16 – CV39) ms. The maximum advised value for CV39 = 12

CV 48 = Step 1
CV 49 = Step 2
…
CV 63 = Step 16

The definition for every step (CV48 .. CV63) is:
- .0 = X2 modulated
- .1 = X3 modulated
- .2 = X4 modulated
- .3 = X5 modulated
- .4 = X2 on
- .5 = X3 on
- .6 = X4 on
- .7 = X5 on

By default the 4 sequencer outputs X2..X5 are prepared to represent phase-shifted gyrolights with a total cycle time of 80ms (CV39 = 6)

4.8.3 Functions

Available functions:
- H = light
- B = brake
- L = blink left
- R = blink right
- F1
- F2
- F3
- F4

The 8 functions each have a Configuration Variable to determine the behaviour:

H (Headlights)

The bits in this parameter have the following function:
- .0 = X0 modulated
- .1 = X1 modulated
- .2 = X2 modulated
- .3 = X3 modulated
- .4 = X0 on
- .5 = X1 on
- .6 = X2 on
- .7 = X3 on

Headlights = CV44, default value = 82 (decimal) = X0 on, X2 on, X1 modulated

B (Brake)

The bits in this parameter have the following function:
.0 = X0 on
.1 = X1 on
.2 = X2 on
.3 = X3 on
.4 = X4 on
.5 = X5 on
.6 = X6 on
.7 = Auto-brake[3]

Brake = CV45, default value = 162 (decimal) = X1 on, X5 on, Auto-brake

L (Left), R (Right)

The bits in this parameter have the following function:
.0 = X0 blink
.1 = X1 blink
.2 = X2 blink
.3 = X3 blink
.4 = X4 blink
.5 = X5 blink
.6 = Y0 blink
.7 = Y1 blink

Left = CV46, default value = 64 (decimal) = blink Y0
Right = CV47, default value = 128 (decimal) = blink Y1

F1

The bits in this parameter have the following function:
.0 = X0 modulated
.1 = X1 modulated
.2 = X2 modulated
.3 = X3 modulated
.4 = X0 on
.5 = X1 on
.6 = X2 on
.7 = X3 on

F1 = CV40, default value = 16 (decimal) = X0 on

[3] Auto-brake means that the brake function depends of the car's speed
Speed = 0: Brake function is activated by the Brake-bit in the communication protocol
Speed > 0: Brake function is activated during slow-down by the Brake-bit in the acceleration parameter

F2

The bits in this parameter have the following function:
- .0 = X0 on
- .1 = X1 on
- .2 = X2 on
- .3 = X3 on
- .4 = X4 on
- .5 = X5 on
- .6 = X6 on
- .7 = Auto-brake Override[4]

F2 = CV41, default value = 128 (decimal) = Autobrake Override

F3, F4

The bits in this parameter have the following function:
- .0 = X0 on
- .1 = X1 on
- .2 = X2 sequencer
- .3 = X3 sequencer
- .4 = X4 sequencer
- .5 = X5 sequencer
- .6 = X6 on
- .7 = Auto-brake Override

F3 = CV42, default value = 8 (decimal) = X3 sequencer (gyrolight)
F4 = CV43, default value = 16 (decimal) = X4 sequencer (gyrolight)

Example configurations

The American Way:

- X0 = headlights
- X1 = tail/brake/blink left
- X2 = tail/brake/blink right
- Y0 = blink left (front)
- Y1 = blink right (front)

CV44 (H) = 22
CV45 (B) = 134
CV46 (L) = 66
CV47 (R) = 132

Ambulance with 4 gyrolights and additional brake

- X0 = headlights
- X1 = tail/brake
- X2 = gyrolight
- X3 = gyrolight
- X4 = gyrolight
- X5 = gyrolight
- X6 = additional brake
- Y0 = blink left
- Y1 = blink right

CV44 (H) = 18
CV45 (B) = 194
CV46 (L) = 64
CV47 (R) = 128
CV42 (F3) = 60

[4] Auto-brake Override means that if Auto-brake is configured in the Brake configuration byte, the brake-function is not automatically activated while braking if the relevant bit in the acceleration parameter is set. This function is intended to leave the brake-light off when the car slows down while going up-hill. If at that moment both the Brake function and ABO functions are activated, the brake light lits at te moment the car actually has come to a stop.

Full featured SUV: headlights high/low beam, fog-headlights, sidelamps, additional brake

- X0 = headlights
- X1 = fog-headlights
- X2 = sidelamps
- X3 = tail/brake
- X4 = additional brake
- Y0 = blink left
- Y1 = blink right

CV44 (H) = 73
CV45 (B) = 152
CV46 (L) = 64
CV47 (R) = 128
CV40 (F1) = 16 (head, low beam)
CV42 (F3) = 1 (head, high beam)
CV43 (F4) = 2 (fog-headlights)

4.8.4 Start-up mode

The behaviour when the decoder is switched-on can be configured by CV38. This CV contains the function-bits to be active when the decoder receives power. This state will remain active until either the first correctly addressed packet is received or sleep mode is entered.

The bits in CV38 have the following function:

.0 = H active
.1 = B active
.2 = L active
.3 = R active
.4 = F1 active
.5 = F2 active
.6 = F3 active
.7 = F4 active

CV38 default value = 13 (decimal) = H + L + R

4.9 Sleep mode and power consumption

If the MCCdec does not receive any valid packet during 100 seconds (also not addressed to other decoders) the decoder will go in sleep-mode. The decoder 'wakes up' as soon as any relevant signal is received. If the 'relevant' signal does not result in reception of an error-free packet (e.g. when the wake-up is caused by EM-noise) the decoder re-enters sleep mode in 4 seconds

Sleep mode is intended to leave cars on the road for a reasonably short period of non-use, e.g. when you shut down your layout one day and intend to continue the next day. Be aware that in order for sleep mode to become active the transmitter needs to be shut down and it is not sufficient to turn of the PC or the communication. In that case the transmitter will keep on sending packets keeping the decoders live.

Power consumption in sleep mode is substantially lower than during operation. The actual current consumption however depends on the battery voltage. The lower the voltage, the higher the necessary current. At a voltage of 2,4V the current in sleep-mode will be roughly 450µA, with a battery capacity of 500mAh therefore around 1.100 hours (starting with full battery). At a voltage of 1,2V the sleep-current is around 900µA

Fig 2 shows the relation between voltage and current consumption in both operational mode and sleep mode. Note that the window of normal operation is between 1,0V and 4,1V

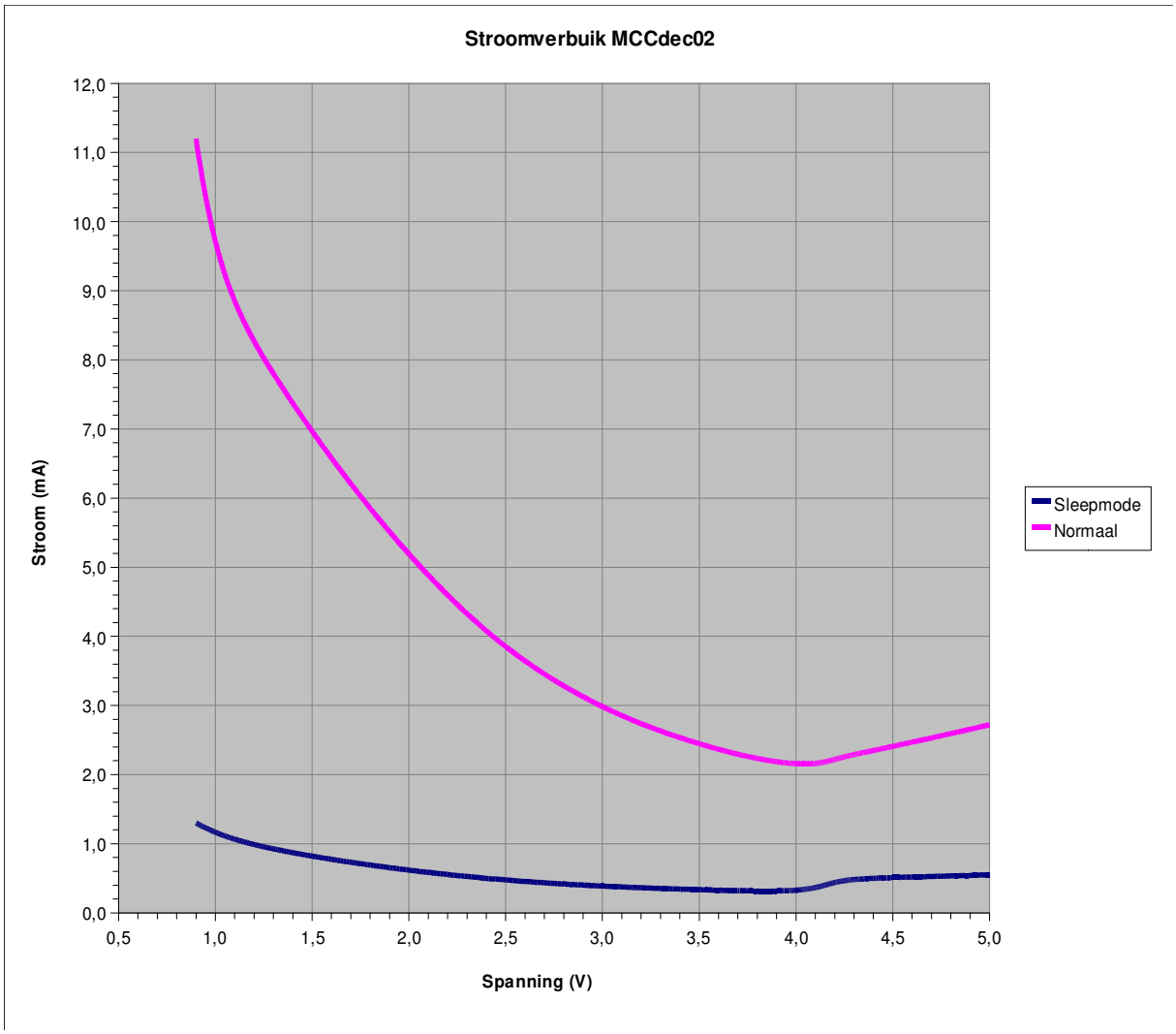

Fig 2: Current consumption vs battery voltage

5 CV Overview

CV	Function	Range	Meaning bits	FS[5]	Comment
0	Factory Reset				Value has no meaning
2	Address-low	0..255		1	Address = 256*CV3 + CV2
3	Address-high	0..15		0	Address = 1..4095
5	Min. battery voltage	0..255		64	
6	“Low Battery” action	0..255	.3..0 = max speed (step) .4 = Alarmlichten .7..5 = vertragingsselector	150	
7	Time-out	0..255		128	
8	Accelerator 0	0..127 (+128)	.0 - .6 = Delay .7 = Brake light	0	
9	Accelerator 1	0..127 (+128)		129	
10	Accelerator 2	0..127 (+128)		130	
11	Accelerator 3	0..127 (+128)		132	
12	Accelerator 4	0..127 (+128)		136	
13	Accelerator 5	0..127 (+128)		144	
14	Accelerator 6	0..127 (+128)		32	
15	Accelerator 7	0..127 (+128)		64	
16	Speed step 0	0		0	
17	Speed step 1	1..255		12	
18	Speed step 2	1..255		24	
19	Speed step 3	1..255		36	
20	Speed step 4	1..255		48	
21	Speed step 5	1..255		60	
22	Speed step 6	1..255		72	
23	Speed step 7	1..255		84	
24	Speed step 8	1..255		96	
25	Speed step 9	1..255		108	
26	Speed step 10	1..255		120	
27	Speed step 11	1..255		132	
28	Speed step 12	1..255		144	
29	Speed step 13	1..255		156	
30	Speed step 14	1..255		168	
31	Speed step 15	1..255		180	
32	PID Control byte	0..243	.0 = EMF input filter .1 = EMF output filter .7..4 = EMF timeout	2	
33	PID P-factor	0..63		16	
34	PID I-factor	0..7		4	
35	PID D-factor	0..63		8	
38	Start-up config.	0..255	See description	13	
39	Sequencer timing	0..12		6	
40	F1 configuration	0..255	See description	16	
41	F2 configuration	0..255	See description	128	
42	F3 configuration	0..255	See description	8	
43	F4 configuration	0..255	See description	16	
44	H configuration	0..255	See description	82	
45	B configuration	0..255	See description	162	
46	L configuration	0..255	See description	64	
47	R configuration	0..255	See description	128	

[5] Factory Setting

CV	Function	Range	Meaning bits	FS[6]	Comment
48	Sequencer step 1	0..255	.0 = X2 modulated .1 = X3 modulated .2 = X4 modulated .3 = X5 modulated .4 = X2 on .5 = X3 on .6 = X4 on .7 = X5 on	18	
49	Sequencer step 2	0..255		3	
50	Sequencer step 3	0..255		3	
51	Sequencer step 4	0..255		33	
52	Sequencer step 5	0..255		3	
53	Sequencer step 6	0..255		6	
54	Sequencer step 7	0..255		6	
55	Sequencer step 8	0..255		14	
56	Sequencer step 8	0..255		12	
57	Sequencer step 8	0..255		72	
58	Sequencer step 8	0..255		12	
59	Sequencer step 8	0..255		132	
60	Sequencer step 8	0..255		13	
61	Sequencer step 8	0..255		13	
62	Sequencer step 8	0..255		9	
63	Sequencer step 8	0..255		11	

[6] Factory Setting

6 Installation

6.1 Introduction

It is assumed that the decoder will be mounted in a car of the Faller Car System® or a car with similar characteristics. As a minimal requirement the car needs to have a battery, a motor, an on/off switch and steering mechanism following a guiding-wire in the road-surface. Cars with rechargeable batteries will have an external connection to charge the batteries. For the remainder of this chapter it is assumed the car is a Faller Car in scale H0 (1:87). For Cars in scale N there may be some limitations because of available space and probably you may want to use smaller LEDs.

The MCCdec02 provides control of the motor, blinkers, brake lights, tail lights, headlights, contour lighting (if applicable) and additional functions. Standard cars are normally not equipped with this kind of lighting, so you'll have to add that yourself. Of course you can do without, but it is not even half the fun.

As headlights for trucks and busses you'd best use white SMD LEDs of size “1206” with a luminosity of at least 300mcd. If you intend to approach the colour of halogen-lamps, choose a “warm-white” type, e.g. with indication “sunny white”. When building smaller cars you'd probably want to use smaller LEDs, e.g. size “805” of “603”.
Tail lights and blinkers can best be made with SMD LEDs in size “603”. This is the smallest size that can reasonably be handled 'by hand', provided you have a good and fine soldering iron (buy a few extra LEDs, you may spoil some in the beginning) For the blinkers, try to find a somewhat 'orange' tint rather than pure yellow.

LEDs have to be operated with series-resistors. The value of these resistors depends on the number of LEDs per output, the type of LED used and the desired intensity. So finding the most optimum resistance may require some experimenting. This variation is exactly the reason why these resistors are not integrated in the decoder.
A reasonable guideline for headlights, brakes and blinkers is a resistance of 470..680Ω at 2 LEDs per output and 330..470Ω with 3 LEDs per output. The brakes are dimmed for tail-light function. When using separate LEDs for tail lights you'll need a somewhat higher resistance (e.g. 1000..2200Ω), otherwise the intensity is too large.
The maximum current an output can source is 20mA. Assuming an operating voltage of approximately 2V for a LED you best use a series resister of at least 100Ω to avoid damage. The easiest is to use axial resistors of 1/8 Watt. These are substantially smaller than the 'standard' 1/4W

If multiple LEDs are connected to 1 output the LEDs are driven in parallel. The output voltage (4V) is too low to drive white or blue LEDs in series. These types require 3..3,5V. You could drive 2 red or yellow LEDs in series, but in that case the series resistor would need to be almost zero, making the current-control very unstable. Therefore, although parallel operation consumes fractionally more power this remains the preferred configuration. As long as you use LEDs of the same brand and type (per output) this should work fine. If different LEDs are driven by a single output provide the different types with its own resistor.
The decoder assumes common-cathode for all LEDs, connected to the – voltage.

6.2 Programming interface

Optionally you can provide a programming interface to the decoder. Determine by yourself whether you consider it to be valuable[7]. Programming is done by a programmer (Microchip

[7] Configuration of CV's can be done 'on the road' without any physical connection. So this interface is really for loading your decoder with new software in case of upgrade or update..

PIC16F88). For reprogramming 5 leads are needed. You can decide yourself how this interface is made physically, however it may be handy to stick to the 'standard' below for some compatibility. In that case it may be possible to upgrade your software e.g. during user group meetings if you don't have a programmer yourself.

The 'standard' is based on a strip of 'turned pin' contacts found in the less-cheap IC sockets. Normally these strips are called 'SIL sockets'. Cut 6 contacts off the strip and close the opening of the second pin, e.g. by soldering. This pin functions as a 'key' to avoid wrong connection. The pin next to the 'key' (outside) is pin 1, pin 2 is closed, the remaining pins are 3, 4, 5 and 6.

The contacts will be given the following functions:
Pin 1: Vpp (programming voltage)
Pin 2: Key (not connected)
Pin 3: - (minus)
Pin 4: PGD (program data)
Pin 5: PGC (program clock)
Pin 6: + (plus)

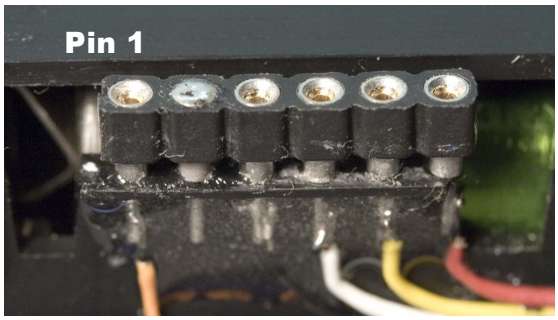

Fig 3: Programming interface

6.3 Preparing the car

Note that you are about to modify your (Faller) car. This process is not completely reversible. If you continue you'll (probably) loose or limit your warranty on your product, perhaps depending on the local law in your country. It does not have to be bad, but be aware of it. Anyway, be sure that you test your original car before continuing this process.

First investigate where and how you can mount the LEDs. Also consider how you can route the wires to the decoder.
Find a place under the car for the receiving coil. The coil should preferably not be higher than 1 cm above the road surface. If the space between the base of the car and road-surface is too small to fit the coil, you may also mount the coil on top of the floor inside the car (assuming the base of the car is made of plastic).[8]

Find a place for the decoder and the resistors. You may solder the series-resistors directly on the connecting points on the decoder if desired.
If you want to add the programming interface, also find a place for that. This can be somewhere on the outside (preferably underneath) the car, but you may also put it somewhere inside if it's accessible without too much trouble. You'll only need it for reprogramming (which should be very rare).

Important Considerations when modifying very small cars:

1) A motor is an electromagnetic device that generates some magnetic fields. How much depends on the construction of the motor. The receiver coil is intended to pick-up magnetic fields from the transmission system and it's function is altered by iron objects. Therefore avoid mounting the receiver coil directly or very close (less than a few mm) to the motor
2) The decoder has a power-converter coil on-board. This coil generates a weak but high-frequency magnetic field which disturbs the function of the receiver coil if mounted very close to each other. If there is no physical alternative than mounting the decoder right on

[8] Place the pick-up coil somewhere in front of the car so it will also stay above the (center of the) road in turns. A good place is just behind the steering mechanism. Try avoiding to place the receiver coil very close to the motor (it may pick-up noise from the motor) or against ferro-metals (which will influence the characteristics of the coil). Permanent magnetism (e.g. from the steering magnet) has no influence on reception

top op the receiver coil, keep the converter coil (the block in the centre of the PCB on the opposite side of the CPU) at the side NOT facing the receiver coil. If the distance is very close you may mount a small brass plate (e.g. 12 x 12 x 0,4mm or so) between the decoder and the receiver coil to minimize interference. Warning: do NOT use iron as 'shield', since this influences the operation of the receiver coil

If you figured out how to do things, continue with the following.

First disconnect the connection to the battery or remove the batteries. Especially rechargeable batteries generate high currents when short-circuited, so be aware that loose wire ends don't touch the battery. Remove the connections from the reed-contact or remove the reed-contact completely. Remove any resistors.

Build in the LEDs, connect them and route the wires to the place where the decoder is going to be. The cathodes of all LEDs connect together. Mark the different wires so you know what is what or use different colored wires.

6.4 Installing the decoder

Decoder Rev02 has 22 connections. Some connection points are 'double' so you don't have to solder multiple wires on 1 connection.
The points marked with 'Batt' in the figures below are internally interconnected (also the one indicated motor+) and therefore are interchangeable. The same counts for the 'GND' poles.

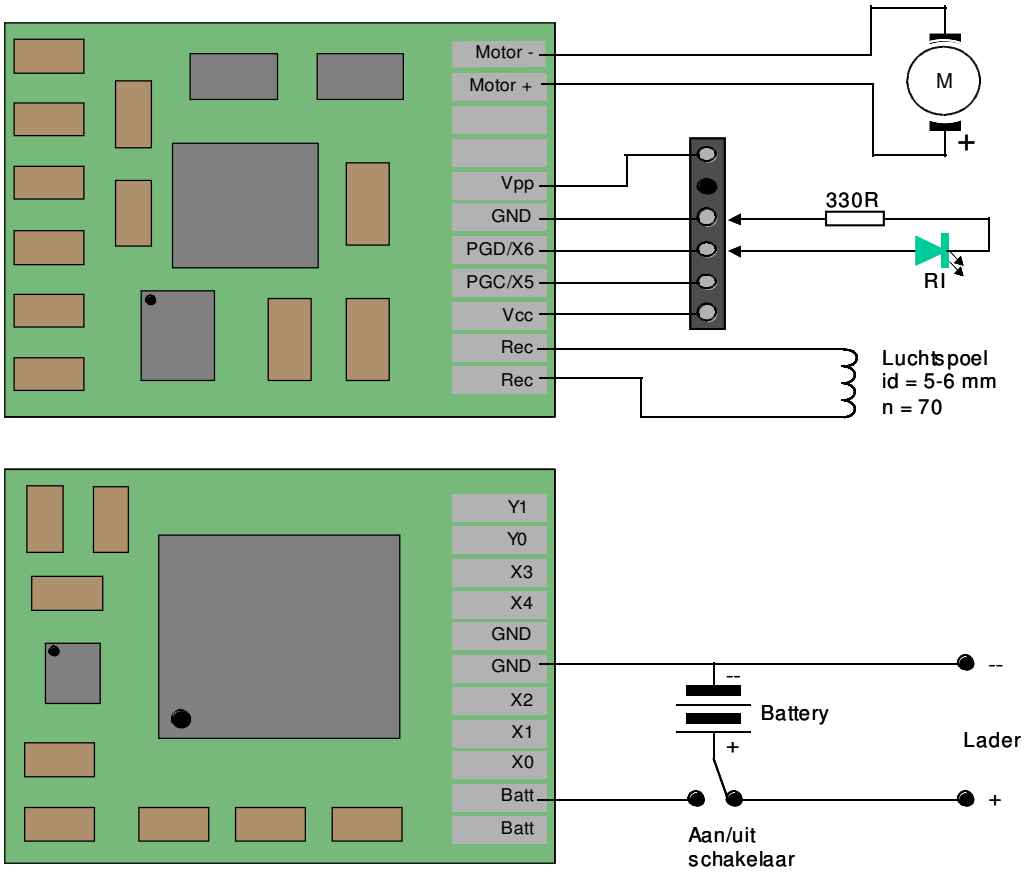

Fig 4: MCCdec 02: Connecting battery, motor, receiver coil and programming interface

If the on/off switch in your car is a 3-pole switch, consider wiring the battery + to the center of this switch. Wire one pole to the charger-pin (+) and the other one to the 'Batt' pole on the

decoder. In this way it is impossible to charge the battery when the decoder is ON, avoiding damage to the decoder when you make a mistake connecting the charger!

Solder a wire to the 'GND' of the decoder for connection to the – of the battery (but do not connect it to the battery yet).

Connect the motor to the decoder as in fig 4 above.

If you want to make a programming interface, do it according to fig 4. If you also intend to use X5 and/or X6, you'd best make a small plug you can put in the programming interface to connect these functions. Otherwise, make sure these functions can be disconnected when the programming interface needs to be used.

Stick the receiver coil underneath (or in) the car and connect it to the decoder as indicated in fig 4.

Important Consideration:

The current through the motor will be substantial in most cases. Since the motor is driven in PWM mode, this current is a high frequency changing current. The current flowing through the wires will generate a changing magnetic field which may, in some cases, become of the same magnitude as the magnetic field from the transmission system. Especially this appears to be the case when using iron-less motors (e.g. 'Faulhaber's).
To avoid interference the best option is to twist the wires carrying the motor-current. These are the 2 wires from the motor to the decoder and the 2 wires between battery and the decoder. In this way the magnetic fields generated by the forward and backward currents eliminate each other almost completely. See fig 5 as an example how it can be done. Also consider that these wires carry current, so don't make them too thin. 'Standard' flexible decoder wire is usually fine.

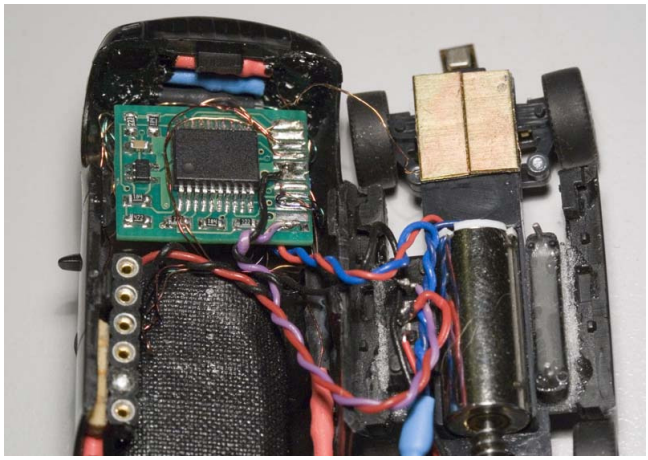

Fig 5: Twisting the wires to avoid high-current interference

Time to connect the LEDs. Connect the anodes of the LEDs via the series-resistors to the decoder as in fig 6. Connect the (common) cathodes to one of the 'GND' points of the decoder.

Note that the given resistor values are indicative. You may have to experiment a bit to find the values you like most. If you want more light, you can go as low as 100Ω per output for yellow and red LEDs and 47Ω for white and blue LEDs. Be aware that if you connect multiple functions to 1 output (e.g. white and red LEDs to the same output, each with their own resistor) the currents add up.

Mount the decoder in the car, e.g. by means of some dual-side adhesive tape. Double-check the connections and finally 'GND' of the decoder to the '-' pole of the battery.

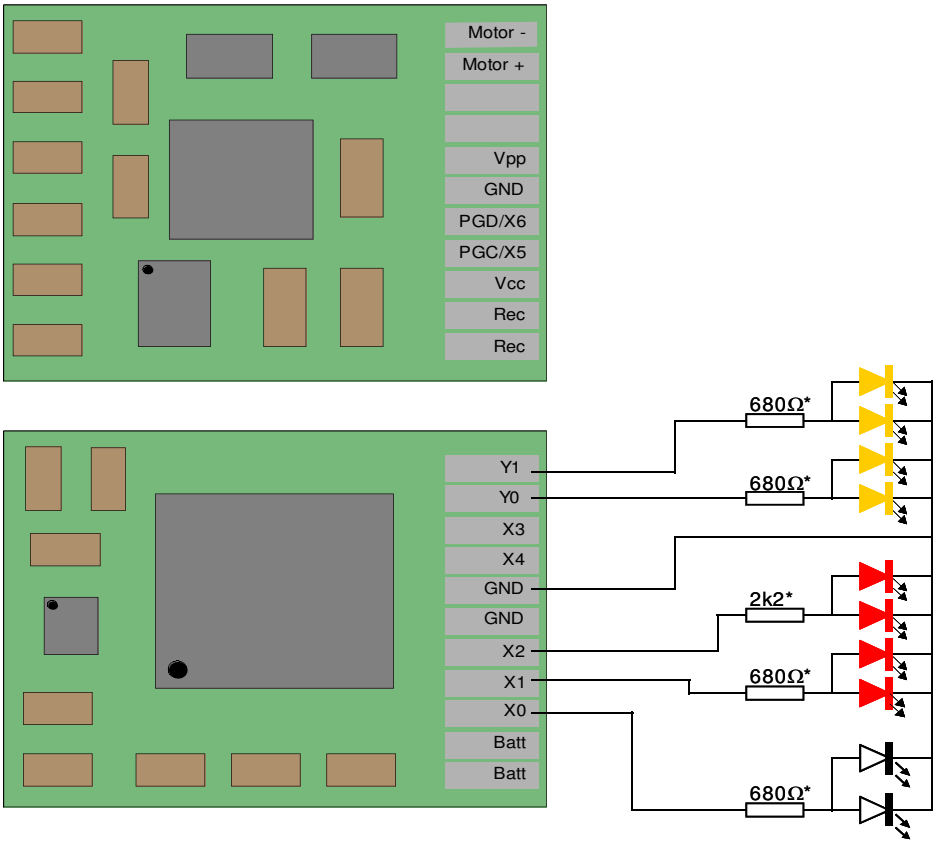

Fig 6: MCCdec 02: Connecting LEDs

6.5 Test and use

If you made no mistakes your car is ready. You can put it on the road, test, configure it to your needs and use it.

WARNING !!
Do not touch the decoder by hand when it is switched-on. The decoder has high-resistance components to minimise current consumption. Touching the decoder may cause the power converter to generate a substantially higher voltage than intended. This is not harmful for you at all, but it will destroy the decoder.

If you power-up the decoder it will enter test-mode. It will blink the blinkers and light head- and tail lights. This start-up mode can be changed by programming the CV's as desired.

If you made the programming interface you get a bonus function: If you connect a LED with series resistor (approx 330Ω) between PGD and – of the programming interface (thus pin 3 and 4) this LED indicates whether the decoder receives a signal. This can help you testing whether reception is fine at all spots on your layout. Note that this receive indicator is combined with X6, so if you use X6 for something else, this 'something' will be activated by the receive indicator function until you reconfigure the functions.

Have fun !