



IRAM-COMP-008

Revision: 1
20 Dec 2003

Contact Author

Institut de RadioAstronomie Millimétrique

CanMot:

Receiver Motor Control Board

Owner Francis Morel (morel@iram.fr)

Keywords:

Approved by:

A.Perrigouard

Date:

August 2004

Signature:

Change Record

REVISION	DATE	AUTHOR	SECTION/PAGE AFFECTED	REMARKS
1	13/06/2003	F. Morel		

Contents

1	General description	3
1.1	The CanMot board	3
2	Hardware.....	3
2.1	The DIP-164 module.....	3
2.2	The ADC	3
2.3	The power driver	4
2.4	The 5V regulator	4
2.5	The CAN connector	4
2.6	The Motor/Potentiometer connector	4
2.7	View of the CanMot board.....	5
2.8	Schematics of the CanMot board	6
3	Software interface.....	7
3.1	Commands and Replies.....	7
3.1.1	Summary of Control and Monitor points:.....	7
3.1.2	Control points in detail:	7
3.1.3	Monitor points in detail:	8
3.2	Real-Time constraints	8
4	Appendix:	8

1 General description

IRAM receivers control uses moving mechanical parts (e.g. backshorts) for fine-tuning. High quality micro DC motors were found to fit perfectly this particular use: They are light, compact, efficient and, when equipped with a 10-revs potentiometer, allow a precise and reliable servo control. There is no need for position indexing, the torque is high, even when the motor is unenergized with shorted rotor, and the position is not lost in case of power shutdown.

In order to control these motors, IRAM developed a CAN interface (named CanMot) which turns each motor into a CAN node. Each motor has a small board (8 cm * 2 cm) attached. This board is connected to external world through a 16-cond flat cable, allowing daisy-chain connection of all motors of one receiver. A 120-Ohm termination resistor must be connected to the last connector of the flat cable.

1.1 The CanMot board

This board uses a very few components. The heart of this board is a Systec module (DIP-164). This very compact (DIP-40 size) module is built around a C164 16-bit microcontroller running at 20 MHz. The module includes a complete CAN interface with its transceivers (NOT opto-isolated), 32 KB RAM, 128 KB Flash, 2 KB SPI EEPROM and some other unused options.

The software was developed in "C" language, using the tools from Keil, supplied with the DIP-164. This firmware resides in the non-volatile Flash-EPROM which can be reprogrammed, if necessary, more than 1000 times.

2 Hardware

The main component is the Systec DIP 40-pin C164-based Microcontroller. The IRAM board is also populated with a low dropout voltage regulator (SO-8), a 12-bit SPI ADC (SO-8) and a power driver (DIP-8).

The 5V regulator supplies the 5V, 80mA for the logics.

The 12-bit ADC is used for position readout (using the 10-rev potentiometer).

The power driver feeds the motor (Max 12V, 200 mA).

A small red LED allows a simple status display. This LED will turn steady ON in case of error, or if the board is in "Stopped" state. It will blink slowly when Requested Position is reached. It will blink fast while the motor is running. More information is available from the Status Register (see below).

2.1 The DIP-164 module

See attached doc (Chapter 4)

2.2 The ADC

A 12-bit sample-and-hold ADC (LTC1286) is used for position readout, converting the potentiometer output voltage into a 12-bit unsigned word. The ADC is a SO-8 SMD chip, and is guaranteed to have no missing code. It is a SPI device. As the native SPI bus of the DIP-164 is already used for the EEPROM and is available on no pins, the SPI protocol for driving the ADC is soft-emulated.

The ADC is able to supply 4000 samples/second. As the potentiometer moves slowly (10 revs = 4096 steps in 100 seconds), a sample rate of 82 samples/seconds might be sufficient. This is not completely true, as the motor has some inertia and does not stop immediately when requested. But this allows oversampling and digital filtering of the ADC output to avoid conversion glitches. This is done calculating the average value of the last 8 samples read from the ADC.

2.3 The power driver

A L272M drives the motor. For dissipation reasons, this device is not a SMD model, but a standard DIP-8. The motor is driven in a bridge configuration, connected between the 2 complementary outputs of the L272M. Motor power requirements are: +/- 12 V, 200 mA Max.

2.4 The 5V regulator

An ADP 3301 is in charge of the 5 V regulation. This very low drop-out voltage SMD SO-8 device is able to supply 5.0 V, 100 mA from a source voltage as low as 5.2 V. The nominal input voltage is 6 V, and this device does not dissipate more than 100 mW.

2.5 The CAN connector

This connector not only connects the CanMot to the CAN-Bus, it also supplies the electrical power needed by the logics and the motor. The board connector is a HE-10 16-pin male connector, which fits the 16-pin WWP HE-10 female connector on the flat cable.

Here is the pinout of the male connector as seen from the board:

PIN 15 +6V M	PIN 13 +6V M	PIN 11 +6V M	PIN 9 +6V L	PIN 7 +6V L	PIN 5 GND	PIN 3 CAN L	PIN 1 GND
PIN 16 -6V M	PIN 14 -6V M	PIN 12 -6V M	PIN 10 +6V L	PIN 8 +6V L	PIN 6 GND	PIN 4 CAN H	PIN 2 GND

Pins description:

GND : Common Ground.

+6V L : + 6V used by logics.

+6V M : +6V used by motor.

-6V M : -6V used by motor.

CAN H : Can High Bus line.

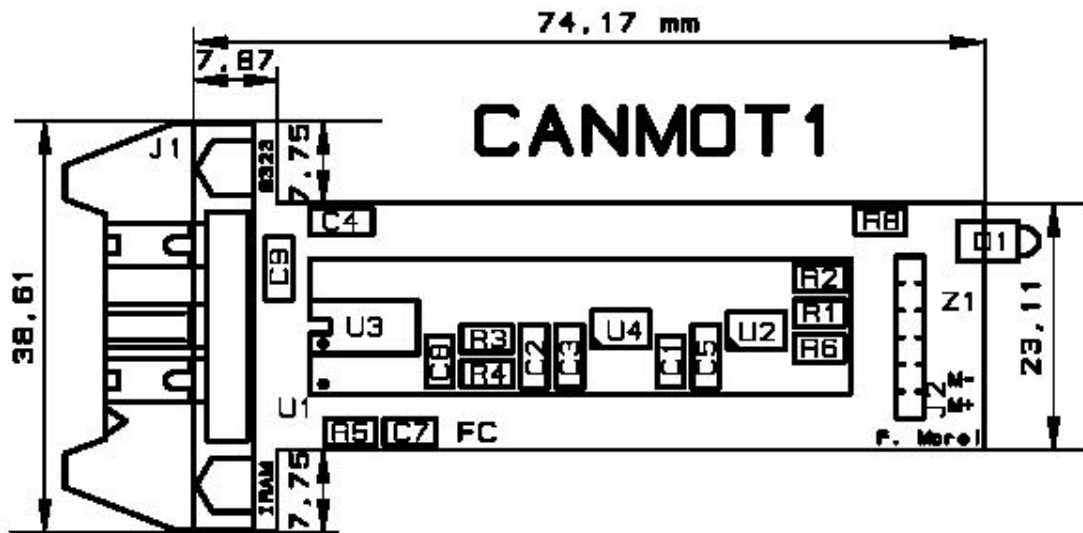
CAN L : Can Low Bus line.

2.6 The Motor/Potentiometer connector

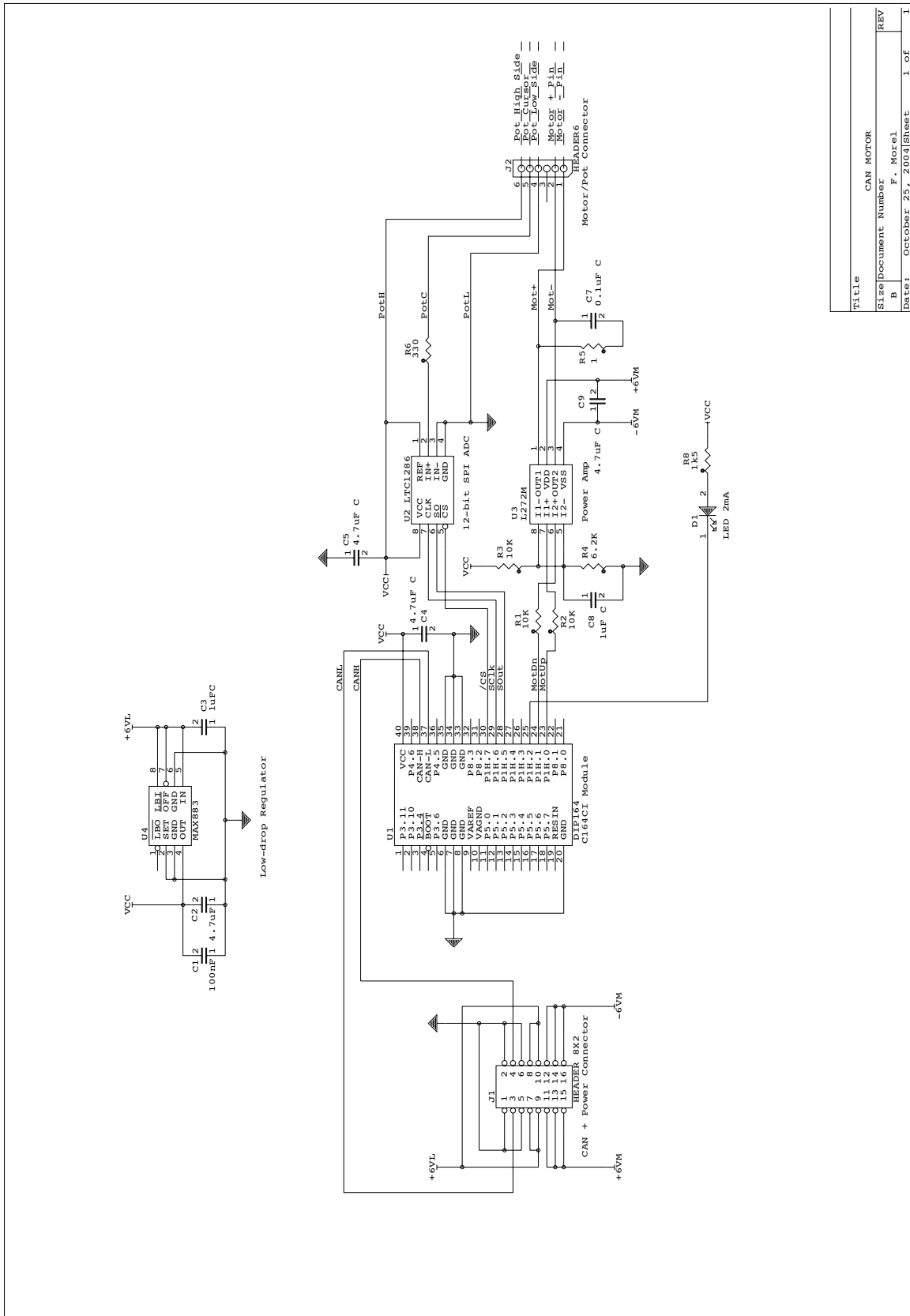
SIL 6-pos connector:

PIN 1	PIN 2	PIN 3	PIN 4	PIN 5	PIN 6
Motor +	Motor -	Unused	Pot L (GND)	Pot Cursor	Pot H (VCC)

2.7 View of the CanMot board



2.8 Schematics of the CanMot board



Title	CAN MOTOR
Size/Document Number	B
REV	P. Morel
Date:	October 25, 2004
Sheet	1 of 1

3 Software interface

The CanMot board accepts commands, generates replies, and executes the requested actions. It does not initiate CAN transactions, and is a slave only. The CanMot board does NOT use remote frames.

Upon start-up or reset, the board gets its 29-bit Node-ID and its 64-bit Serial Number from an internal EEPROM. If the EEPROM is empty or faulty, the board will use default ID "40000H". The EEPROM contents will not be altered if a new version of the firmware is downloaded into the Flash EPROM.

3.1 Commands and Replies

-*Read Actual Position*: The board replies sending the actual position of the motor (unit = 1 ADC step).

-*Set Requested Position*: If this command is accepted, the Requested Position becomes the new target position and the motor starts immediately. The position is expressed in ADC steps. As the ADC is a 12-bit one, the position range is 0 to 4095(decimal). The potentiometer used for position readout is a 10-revolutions model, and thus, one ADC step results in a 0.87 Degree rotation of the axis. For security reasons, this range is restricted to [7 — 4088].

If the Requested Position is out of range, the motor stops, and an error is reported in the Status Register.

-*Read Status*: The board replies sending an encoded status (see below).

-*Stop*: The motor stops immediately.

-*Reset*: The board is reset. This command is equivalent to shutdown/restart.

-*Broadcast Message*: Upon receipt of a message with ID = 0x0, the CanMot board sends a reply message, using its Node_ID, and its 64-bit serial Number as 8-byte data.

3.1.1 Summary of Control and Monitor points:

Name	CAN ID	Data Size	Description
SET_MOT_RPOS	Node-ID + 0x101	2	Set Requested Position
SET_MOT_STOP	Node-ID + 0x103	1 (dummy)	Stop
SET_MOT_RESET	Node-ID + 0x1FF	1 (dummy)	Reset
GET_MOT_APOS	Node-ID + 0x100	3	Read Actual Position
GET_MOT_STS	Node-ID + 0x102	4	Read Status

3.1.2 Control points in detail:

Name	SET_MOT_RPOS
CAN ID	Node-ID + 0x101
Description	Sets the Motor Requested Position
Data	2 bytes Data bytes[0..1] = 12-bit unsigned value of Actual Position. byte[0] = Position MSByte, byte[1] = Position LSByte.

Name	SET_MOT_STOP
CAN ID	Node-ID + 0x103
Description	Stops the Motor.
Data	1 dummy byte

Name	SET_MOT_RESET
CAN ID	Node-ID + 0x1FF

Description	Resets the Motor. This command has highest priority, and executes inside the CAN interrupt routine.
Data	1 dummy byte

3.1.3 Monitor points in detail:

Name	GET_MOT_APOS
CAN ID	Node-ID + 0x100
Description	Reads the Motor Actual Position
Data	3 bytes Data bytes[0..1] = 12-bit unsigned value of Actual Position. byte[0] = Position MSByte, byte[1] = Position LSByte. Transaction report in byte[2]: Bit[0] = CAN Warning.

Name	GET_MOT_STS
CAN ID	Node-ID + 0x102
Description	Reads the Motor Status
Data	4 bytes Data byte[0] = 1-bit Status Code <i>0x20: Board reset</i> <i>0x10: Board stopped</i> <i>0x8: Requested Position error (bytes[1..2] = Requested Position)</i> <i>0x4: Position Aborted (bytes[1..2] = Actual Position)</i> <i>0x2: Position Reached (bytes[1..2] = Requested Position)</i> <i>0x1: Running (bytes[1..2] = Actual Position)</i> Data byte[1..2] = (Requested OR Actual) Position. byte[1] = Position MSByte, byte[2] = Position LSByte. Transaction report in byte[3]: Bit[0] = CAN Warning.

3.2 Real-Time constraints

Only one interrupt is used, the CAN interrupt. This interrupt is triggered upon reception of a CAN message with matching ID. The CAN command is then stored into a buffer, and will execute within 250 microseconds. The size of the buffer is 64 messages. This will guarantee the maximum priority to CAN commands.

The motor runs at full velocity, except when close (+/- 2 steps) to the final position. In such a case, it runs at half-velocity only, in order to improve the final positioning (within ½ LSB). The velocity command uses PWM (Pulse Width Modulation at 1 kHz), always allowing maximum torque.

A velocity check is also implemented on the CanMot board: If, for any reason, the motor does not move as requested (wrong direction or no move at all), the board will cancel the command after a 1 second delay and report "Position Aborted" in the Status register.

4 Appendix:

See doc of the DIP164: <\\netapp1\computer\doc\can\canMot\dip164.pdf>

