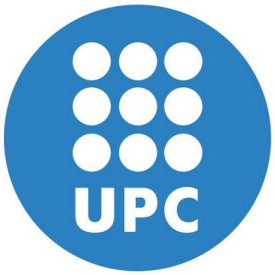
REDESCUBRIENDO LA PLATAFORMA ROBÓTICA EXPERIMENTAL MASHI. ANEXOS

JOAQUÍN CORTÉS FUENTES



Grau en Enginyeria en Tecnologies Industrials
Enginyeria de Sistemes, Automàtica i Informàtica Industrial (ESAII)
Universitat Politècnica de Catalunya
Escola Tècnica Superior d'Enginyeria Industrial de Barcelona (ETSEIB)

Enero 2017

SUPERVISORES:
Cecilio Angulo Bahón
ENTREGADO:
Enero 2017

Índice general

A Componentes

A.1. Motores y ruedas

Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

Motor Mount and Wheel Kit (#27971) with Position Controller (#29319)

Features

- Powerful DC motors for plenty of torque
  ~150 RPM @ 12.0 VDC, 1.50 A, no load
  ~190 RPM @ 14.5 VDC, 1.60 A, no load
- Precision machined 6061 aluminum hardware
- Conveniently positioned screw holes make mounting this kit a breeze.
- Rugged pneumatic tires are well-suited for a variety of terrains
- +5 volt Supply for Position Controllers
- 36 encoder positions per revolution; approx 0.5 inch resolution with 6" wheel
- Compatible with any microcontroller
- Single I/O line can control up to 4 Position Controllers
- Strong but light: only 3.2 lbs (1.45 kg) per wheel assembly (kit contains two).

Application Ideas

- Sturdy and stylish midsize robot platforms
- Accurate distance tracking
- Autonomous exploration and data collection
- Part of a robust navigation system when incorporated with GPS and compass modules

General Description

The Motor Mount and Wheel Kit combines powerful 12 VDC motors and precision machined aluminum hardware to provide the power, strength, and beauty demanded by midsized robots. All custom parts are CNC machined at Parallax headquarters in Rocklin, California from 6061 billet aluminum. Conveniently positioned screw holes on the top, bottom, and front face of the bearing blocks provide a variety of quick and easy mounting options. The included 6 inch (15.3 cm) pneumatic rubber tires are durable enough to handle a variety of smooth or rugged terrains without hesitation.

The Position Controllers use a quadrature encoder system to reliably track the position and speed of each wheel at all times. With the included plastic encoder disks, each Position Controller has a resolution of 36 positions per rotation; this equates to approximately 0.5 inches of linear travel per position using the included 6 inch tires. The Position Controllers calculate and report position and average speed data on command. This leaves the main processor free to handle more important tasks like reading GPS coordinates, processing sensor information, and maneuvering complex environments.

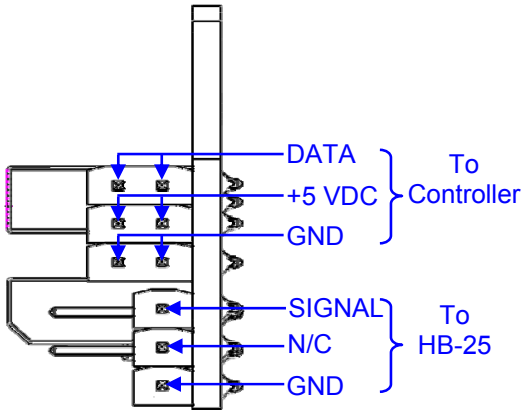
The Position Controllers are compatible with any microcontroller via a single-wire half-duplex asynchronous serial communication (UART) bus. Up to four Position Controller devices can be controlled on the same bus to minimize I/O requirements.

For increased functionality, Position Controllers can be interfaced with HB-25 motor controllers (#29144; sold separately) to control the position of the wheel and automatically provide smooth speed ramping as well as accurate position advancement capability.

Position Controller Device Information

Pin Description

Pin Name	Function
DATA	Communication line used to send and receive data – Connect to single-wire UART bus
+5V	Regulated DC Supply Voltage (V_{CC})
GND	Common Ground
SIGNAL	Pulse output line to control an optional HB-25 motor controller



Absolute Maximum Ratings

NOTICE: These values are the limit before permanent damage may occur, but proper operation of the device is not guaranteed all the way up to these values. See the "Electrical Characteristics" section for recommended operating conditions for the Position Controller.

Parameter	Min.	Max.	Units
Operating Temperature[1]	-55	125	°C
Storage Temperature[1]	-65	150	°C
Voltage on Data Pin (with respect to Ground)[1]	-0.5	V_{CC}+0.5	V
Maximum Operating Voltage[1]	--	6.0	V

Notes:
(1) These values are taken from the Atmel ATtiny2313 device documentation.

DC Electrical Characteristics

Ambient Temperature = -40°C to 85°C

Parameter	Min.	Typ.	Max.	Units
Operating Voltage (V_{CC})	4.5	5.0	5.5[1]	V
Average Power Supply Current (I_{CC})	--	50	--	mA
Ground (GND)	--	0	--	V
Data Pin Input Low Voltage[1][2]	-0.5	--	$0.3V_{CC}$	V
Data Pin Input High Voltage[1][2]	$0.6V_{CC}$	--	V_{CC} + 0.5	V
Signal Pin Output Low Voltage[1]	--	GND	0.7[3]	V
Signal Pin Output High Voltage[1]	4.2[3]	V_{CC}	--	V

Notes:
(1) These values are taken from the Atmel ATtiny2313 device documentation.
(2) V_{CC} = 4.5 V to 5.5 V
(3) V_{CC} = 5 V; I_{PIN} = 20 mA

Theory of Operation

The Position Controller is designed to manage certain operations associated with wheel motion that traditionally consumed system resources, processing power, and execution time in the main microcontroller. By using the Position Controller to manage these functions instead, the main microcontroller is free to focus on more important tasks.

Each Position Controller has an on-board microcontroller which continually tracks the position and average speed of the wheel. In conjunction with the plastic encoder disk, two optical interrupter switches generate a quadrature encoded waveform which is processed by the microcontroller. Based on the rate and specific sequence of the pulses, it is possible for the Position Controller to determine how fast, and in which direction, the wheel is turning.

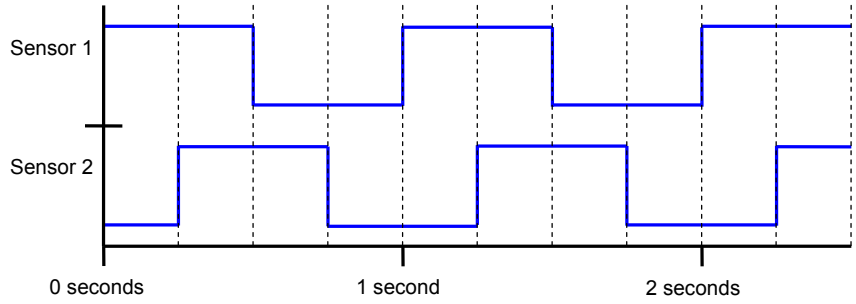


Figure 1

Quadrature encoded voltage waveforms for a wheel rotating at 4 positions per second. Notice that rising edges occur first on Sensor 1, signifying that the wheel is rotating in the positive direction.

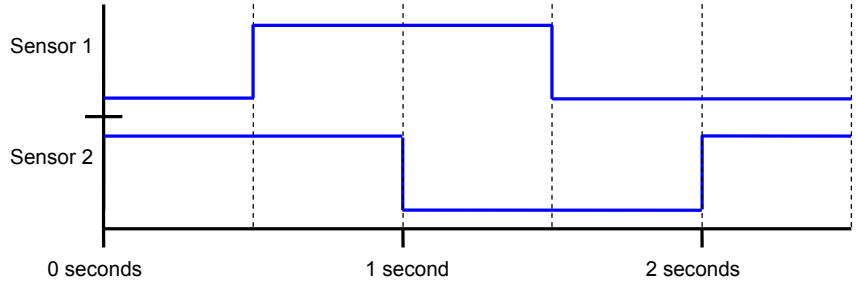


Figure 2

Quadrature encoded voltage waveforms for a wheel rotating at 2 positions per second. In this case, rising edges are seen on Sensor 2 first, signifying a negative direction of rotation.

The current position value appropriately increments or decrements by 1 each time there is a waveform transition. The current average speed value is updated every 20 ms and is an accumulated average over the previous 0.5 seconds. It is important to note that this is an average speed value, not the instantaneous speed of the wheel. See the Command Set Details section on page 12 for more information.

The Position Controller automatically generates the appropriate pulses to drive an HB-25 motor controller (#29144; sold separately). See Interfacing with the HB-25 Motor Controller on page 11 for more information. This configuration dramatically increases the Position Controller's functionality since it gains control over the wheel's position and speed rather than simply measuring them. The Position Controller stores the desired location as a set point which it continually seeks. If the wheel is rotated away from the current set point, it increases power to the motor in the opposite direction in order to return back to the original position. Additionally, when the wheel is being driven on an incline or decline, the Position Controller automatically increases or decreases power to the motor to maintain the true desired speed.

The position advancement scheme is designed to move the wheel a user-set distance (TRVL command) at a user-set maximum speed (SMAX command) with smooth user-set acceleration and deceleration (SSRR command) along the way. Each variable has a default value when powered on, but can be directly modified with specific commands (see the Command Set Details section on page 12 for more information). A typical speed profile for traveling a certain distance is shown in Figure 3 below.

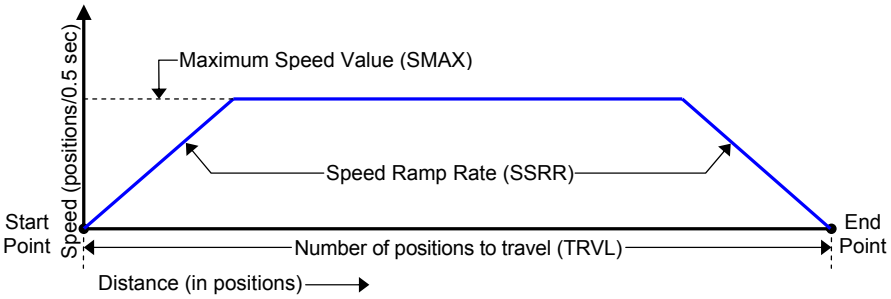


Figure 3

Typical speed profile for traveling a certain distance. The set point is advanced incrementally until it reaches the endpoint.

When the wheel advances toward the end point, it is really the set point that is being incrementally advanced. Higher speeds are achieved by advancing the set point by a greater distance each increment than for lower speeds. Since the Position Controller constantly drives the wheel to match its set point, it is able to achieve the true overall user-set speed.

The Position Controller accelerates at the user-set speed ramp rate until it reaches the user-set speed. Then it uses the current average speed to calculate the distance required to decelerate smoothly to the final end point without overshoot. When traveling in one direction, if a new position advancement value requires the wheel to reverse direction, it will still decelerate smoothly to a stop at the user-set ramp rate before re-accelerating in the opposite direction toward the new end point.

The main microcontroller communicates easily with up to four Position Controllers over a single-wire universal asynchronous serial receiver and transmitter (UART) bus. The UART operates at 19.2 kbits/sec data rate, 8-bit frame, 1 stop bit, no parity. See the Module Schematic section on page 18 for details on how the bus is electrically connected. In most cases, the data line on the Position Controller can be directly connected to the main microcontroller with no additional external components required.

Setting the Device ID

When more than one device is connected on the same bus, it is necessary to give each device a unique device ID. Each device's ID value is physically set by its corresponding jumper configuration read when powered on (see table below). Notice that the possible values are 1 through 4. Commands can address ID 0 as a special case address to send commands to all devices at once. Devices are shipped with both jumpers installed (ID value 1).

A|B	ID Value*	Binary
▪▪	1	001
▪□	2	010
□▪	3	011
□□	4	100

* ID Value 0 is reserved for addressing all devices at once.

Interfacing with the HB-25 Motor Controller

The Position Controller is designed to be interfaced with an HB-25 motor controller to control the position and speed of the motor and wheel. This can be accomplished by connecting a standard three-pin cable from the header pins marked "To HB25" to the three-pin header on the HB-25. Be sure to connect each end of the cable in the proper orientation of white "W", red "R", and black "B" as marked on both the Position Controller and the HB-25. Connect the power leads on the HB-25 as usual (positive supply to the "+" and ground to the "—").

When using the Motor Mount and Wheel Kit in the standard mirrored left motor and right motor configuration, connect the yellow lead of each motor to "M1" and the blue lead of each motor to "M2" on each respective HB-25. In this configuration, the orientation of the Position Controller on the right-hand motor must be reversed in software (see the SREV command). When wired and programmed this way, the motors will rotate in opposite directions, but the direction of rotation will be interpreted as "positive" by both Position Controllers. The result is a straight path of the vehicle (rather than rotating on center).

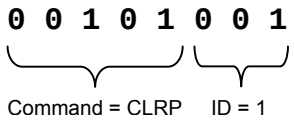
After wiring both leads identically, and setting the Position Controller on the right to be reversed orientation, both Position Controllers can be given a command to travel forward and will correctly travel in the same direction. If one of the motors starts traveling away uncontrollably (when they should otherwise be holding a position), it is likely an indication that the motor wiring is incorrect, or that the command to reverse the Position Controller's orientation was sent to the wrong Position Controller, or not sent at all.

Command Set Summary

The Position Controller's command set is grouped by category as query commands, configuration commands, and action commands. Query commands request information from a specific individual Position Controller board and generate a reply. Action commands affect the output of the Position Controller when it is being used to control an HB-25 motor controller, and do not generate a reply. Configuration commands set certain parameters in one or more Position Controller boards and also do not generate a reply. Any variables modified by configuration commands are reset to default when the device loses power.

Command codes are byte-sized with the upper 5 bits containing the command value and the lower 3 bits containing the ID value of the device for which the command is intended, minimizing the required byte count for each command.

Command Code Byte Example



Depending on the command, a command code may require zero, one, or two additional bytes of data. For commands which require two bytes of data, or commands which generate a reply, data is sent high byte first, then low byte. The Command Set Details section below contains thorough information for each command.

A Position Controller will accept a command if the ID value sent in the command code matches the Position Controller's own ID. Additionally, a Position Controller will also accept configuration commands and action commands when the ID value is 0. This is a special case ID value which allows certain commands to be sent to all devices present on the bus at once. Note that query commands do not generate a response when the special case ID value of 0 is used, since doing so would likely cause bus contention between devices.

Command Set Summary Table

	Description	Command		Hex	Binary	Single	All *
Query	Query Position	QPOS	1	0x08	00001 000	●	
	Query Speed	QSPD	2	0x10	00010 000	●	
	Check for Arrival	CHFA	3	0x18	00011 000	●	
Action	Travel Number of Positions	TRVL	4	0x20	00100 000	●	●
	Clear Position	CLRP	5	0x28	00101 000	●	●
Configuration	Set Orientation as Reversed	SREV	6	0x30	00110 000	●	●
	Set Tx Delay	STXD	7	0x38	00111 000	●	●
	Set Speed Maximum	SMAX	8	0x40	01000 000	●	●
	Set Speed Ramp Rate	SSRR	9	0x48	01001 000	●	●

* Every command can be used to address a single Position Controller device; additionally, configuration commands and action commands can be used to address all devices on the bus at once by using the special case ID value 0.

Note that the binary value for each command is simply the command number shifted left by 3 places. It is usually most expedient to store the pre-shifted binary/hex command values in a table of constants. Generating the correct command code byte can be accomplished by summing or logically OR-ing the pre-shifted command constant with the ID value of the desired device.

Command Set Details

QPOS – Query Position

Command #	Hex Value	Binary Value	Command Category/Type	
1	0x08	00001 000	Query	Single

Transmit Data Format
[Command:Address]

Receive Data Format
[ValueH],[ValueL]

Description: Returns the current position as a 16-bit signed value. The value is returned high byte first (ValueH) then low byte (ValueL).

The position value increments by 1 for each position traveled in the positive direction, and decrements by 1 for each position traveled in the negative direction. By default, positive direction is defined as counter-clockwise rotation when observing the wheel from the outside (screw side of the Position Controller board). When the sensor orientation is reversed (see the SREV command), the definition of positive direction is reversed.

The position value is cleared to 0 when the device is powered on. Additionally, the position value can be cleared using the Clear Position (CLRP) command. Note that positive advancement beyond +32,767 results in a rollover to -32,768.

QSPD – Query Speed

Command #	Hex Value	Binary Value	Command Category/Type	
2	0x10	00010 000	Query	Single

Transmit Data Format
[Command:Address]

Receive Data Format
[ValueH],[ValueL]

Description: Returns the current average speed in positions/0.5 sec. This is a signed 16-bit number. The current speed value is updated every 20 ms and is an average over the previous 0.5 second. The value is returned high byte first (ValueH) then low byte (ValueL).

It is important to note that this is an average speed value, not the instantaneous speed of the wheel. The difference is most noticeable when the speed changes abruptly. For example, if a speed of 10 positions/0.5 second instantly increases to 30 positions/0.5 second, the average speed value will immediately start increasing, but will take up to 0.5 second to register the new actual speed. This can be beneficial when traveling over uneven terrain since the overall average speed is usually more meaningful than a fluctuating instantaneous speed reading.

CHFA – Check for Arrival

Command #	Hex Value	Binary Value	Command Category/Type	
3	0x18	00011 000	Query	Single

Transmit Data Format
[Command:Address],[Tolerance]
{0 ≤ Tolerance ≤ 255}

Receive Data Format
[Value]
{Value = 0xFF (True); Value = 0x00 (False)}

Description: Checks to see if the device has arrived at its destination within a specified position tolerance. The position tolerance value can be 0 to 255, and must be specified in the byte directly following the command code. This function will return "True" (0xFF) if the current average speed is zero, and the current position matches the end point within ± the position tolerance value. Otherwise this function will return "False" (0x00).

TRVL – Travel Number of Positions

Command #	Hex Value	Binary Value	Command Category/Type	
4	0x20	00100 000	Action	Single, All

Transmit Data Format
[Command:Address],[NumberH],[NumberL]
{-32768 ≤ Number ≤ 32767}

Receive Data Format
N/A

Description: This command advances the end point by "Number" of positions. Number is a signed 16-bit value sent high byte first (NumberH) then low byte (NumberL). Any time the end point is moved away from the current set point, the Position Controller will accelerate at the speed ramp rate (see SSRR command) to the set maximum speed (see SMAX command) and automatically decelerate just in time to stop at the exact end point.

TRVL commands are accumulative. For example, sending a command to travel +200 positions followed immediately by another command to travel +130 positions is equivalent to sending a single command to travel +330 positions. Likewise, a command to travel +200 positions followed by another command to travel -130 positions is equivalent to a single command to travel +70 positions.

It is possible to send a TRVL command which requires the wheel to reverse its current direction of travel in order to seek the end point. When this occurs, the wheel will decelerate at the speed ramp rate in its current direction before re-accelerating in the opposite direction. This effect is also seen when a TRVL command moves the end point to a position which is still in the same direction of travel but is too close to be reached by decelerating at the speed ramp rate. Rather than stopping abruptly or decelerating faster than the speed ramp rate will allow, the wheel will decelerate smoothly past the end point to a stop before re-accelerating in the opposite direction to the current end point.

Sending a command to travel 0 positions is a special case used to bring the wheel to a smooth stop when traveling. The current speed is decelerated at the speed ramp rate and the end point is advanced to the exact location where the speed will reach zero. Note that this special case is not considered accumulative and will override the previously remaining distance to travel. However, TRVL commands sent after this will be accumulative as expected.

To immediately stop the wheel (without decelerating at the speed ramp rate) the user should send a Clear Position (see CLRP command) which effectively halts the wheel at its current position.

CLRP – Clear Position

Command #:	Hex Value:	Binary Value:	Command Category/Type:	
5	0x28	00101 000	Action	Single, All

Transmit Data Format	Receive Data Format
[Command:Address]	N/A

Description: This command resets the current position, end point, and set point back to zero. This effectively halts the wheel at its current position. The current position becomes the new starting point for position measurement and motion; and consequentially clears any TRVL commands which may be in progress.

This command can also be used as a means to "soft-reset" a Position Controller. For example, if the main microcontroller is reset while in the middle of sending a TRVL command, any data received after that will be interpreted as valid data, causing the wheel to begin traveling to an unpredictable location. By sending the CLRP command three times in a row to all present devices, it is ensured that any position advancement will be cleared; however all configuration data will remain unchanged. To completely reset a Position Controller to default including configuration data, it is recommended to cycle power. This is considered a "hard-reset".

SREV – Set Orientation as Reversed

Command #	Hex Value	Binary Value	Command Category/Type	
6	0x30	00110 000	Configuration	Single, All

Transmit Data Format	Receive Data Format
[Command:Address]	N/A

Description: Reverses the default definition for positive direction of travel. This is used when the orientation of a sensor is physically reversed. For example, in robotic applications where there is a left wheel and a right wheel, when the robot is traveling forward one wheel is traveling in the positive direction and one in the negative direction (when looking at each wheel from the side). By sending the SREV command to one of the Position Controllers, they can both be viewed as traveling in the positive direction.

The default definition for positive direction of travel is counter-clockwise rotation when observing the wheel from the outside (screw side of the Position Controller board). After receiving the SREV command, a Position Controller will interpret positive direction of travel as clockwise rotation from the same viewpoint.

This setting becomes active immediately and is persistent while the device is powered. That is, sending this command more than once does not toggle the definition for positive direction of travel; and the setting only returns to default after the device loses power.

STXD – Set Tx Delay

Command #	Hex Value	Binary Value	Command Category/Type	
7	0x38	00111 000	Configuration	Single, All

Transmit Data Format	Receive Data Format
[Command:Address],[Delay] {0 ≤ Delay ≤ 255}	N/A

Description: Allows the user to specify the minimum delay period to wait before a Position Controller will respond to a query command and between bytes of data. Some microcontrollers like the Basic Stamp cannot receive data immediately after sending data. The Basic Stamp 2 (BS2) for example, needs at least around 330 µs to complete a SEROUT command and initialize a SERIN command before it can successfully begin receiving data. Other microcontrollers may require a longer delay or no delay at all. The minimum delay period can be calculated as follows:

*minimum delay period = (delay value * 4.34 µs) + 40 µs*

Note that the delay period calculation above is the minimum delay that the Position Controller will wait before sending responses. The actual delay may be slightly longer. The user is encouraged to experiment with different delay values until an optimal value is found for that system. The default delay value on power up is 115 which equates to about 540 µs. This delay is plenty when communicating with a Position Controller using a BS2.

Note that for query commands which return data, the data value is acquired immediately when the command is received; but is just delayed from being sent back. This may be important since the data received back could be over 2 ms old by the time the two bytes are sent using the maximum delay value of 255.

SMAX – Set Speed Maximum

Command #:	Hex Value:	Binary Value:	Command Category/Type:	
8	0x40	01000 000	Configuration	Single, All

Transmit Data Format
[Command:Address],[SpeedH],[SpeedL]
{0 ≤ Speed ≤ 65535}

Receive Data Format
N/A

Description: Sets the maximum desired rate of travel in units of positions/0.5 second (same units as the QSPD command). The speed value is an unsigned 16-bit value sent high byte first (SpeedH) then low byte (SpeedL). The default value is 36 positions/0.5 sec which equates to 120 rotations/minute. Changes in the maximum speed limit become effective immediately. It is recommended that the maximum speed value only be modified during a stopped state; however, the SMAX command technically allows for a new maximum speed to be set at any time.

The maximum speed value is considered a "true" speed because the Position Controller will maintain a constant overall average speed regardless of terrain, slope, or motor loading (as long as the driving motor has sufficient power to do so). Because the Position Controller can constantly monitor the wheel's actual speed, it can dynamically increase or decrease power to the motor in order to maintain the desired speed value. For example, if the motor is driving up a hill, the Position Controller will throttle up power to the motor to maintain constant speed; and will throttle down or even apply an opposite torque if necessary when traveling down hill.

Note that at very low speeds (around 1 position/0.5 second) motion may not be as smooth. The user may notice the individual advancements of the set point every 0.5 second; however, the overall rate of travel will still exactly match the user-specified maximum.

*While not recommended, it is possible to change the maximum speed value in mid-travel. Keep in mind that this value is the maximum speed that the position controller is allowed to travel. During motion, if the Position Controller receives a new speed maximum which is greater than the current speed maximum, it will begin accelerating at the speed ramp rate up to the new speed maximum. If the Position Controller receives a new speed maximum which is lower than the current speed maximum, the speed will instantly be clipped to the new value which will generally be abrupt and undesirable.

SSRR – Set Speed Ramp Rate

Command #:	Hex Value:	Binary Value:	Command Category/Type:	
9	0x48	01001 000	Configuration	Single, All

Transmit Data Format
[Command:Address],[Rate]
{1 ≤ Rate ≤ 255}

Receive Data Format
N/A

Description: This command sets the rate of acceleration/deceleration for the beginning/end of travel. The speed ramp rate is an 8-bit unsigned value with units of positions/0.25 sec^2. This value is the rate of change of speed (in positions/0.5 sec) per 0.5 second. The default value on power up is 15 positions/0.25 sec^2.

When a command is given to travel a number of positions, the Position Controller will begin accelerating at the speed ramp rate until it reaches the maximum speed value, or until it must begin decelerating to reach the end point without overshooting; whichever comes first.

A ramp rate of 0 is not valid since the Position Controller would never be allowed to increase its speed in order to begin traveling. Consequentially, if a speed ramp rate of 0 is received, it will automatically be increased to the minimum value of 1.

The speed ramp rate should only be modified while stopped. Attempting to modify this value while traveling will produce undesirable results.

Communication Protocol

The main microcontroller communicates easily with up to four Position Controllers over a single-wire TTL-level UART bus. The UART operates at 19.2 kbits/sec data rate, 8-bit frame, 1 stop bit, no parity. When a Position Controller is not actively occupying the bus to transmit data, it remains a high-impedance input.

Each command may require zero, one, or two additional bytes of data (see the Command Set Details section). When a certain command generates a reply, the Position Controller will wait at least the minimum transmit delay period before sending back data, as well as between bytes of data. When receiving a command, Position Controllers will wait indefinitely for the required number of bytes. This is true even when the command is not addressing the specific device's ID since the device will discard the same expected number of bytes for a command before listening for new commands.
Note that query commands should never be sent to a device ID which is not present on the bus. Since the other Position Controllers will discard the expected number of bytes based on the command, it is possible for the Position Controllers to become unsynchronized with the main microcontroller.

Figure 4 below shows an example of successful command/data exchange using the Query Speed command with a Position Controller device ID of 2.

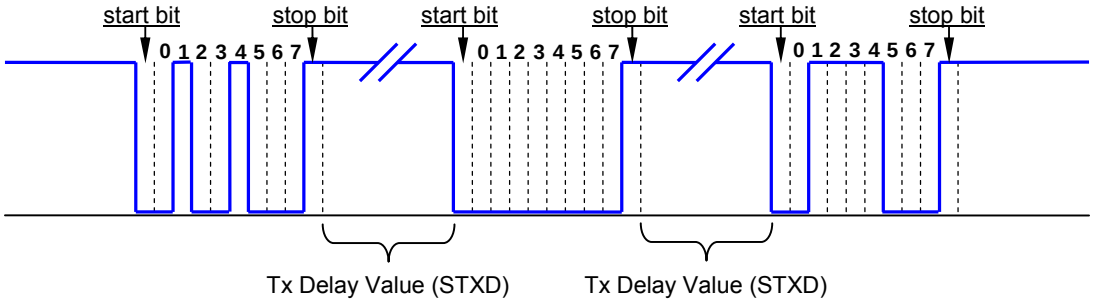
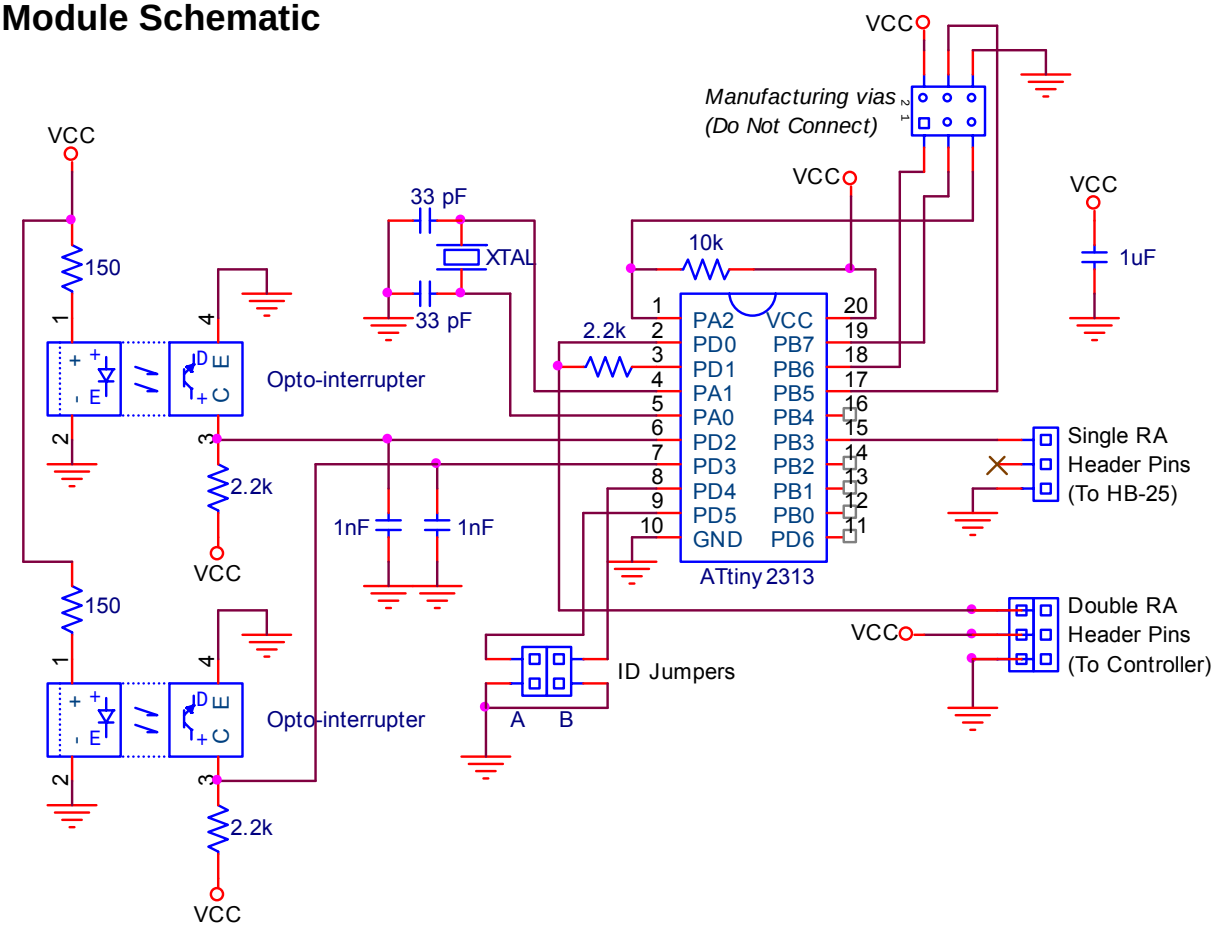
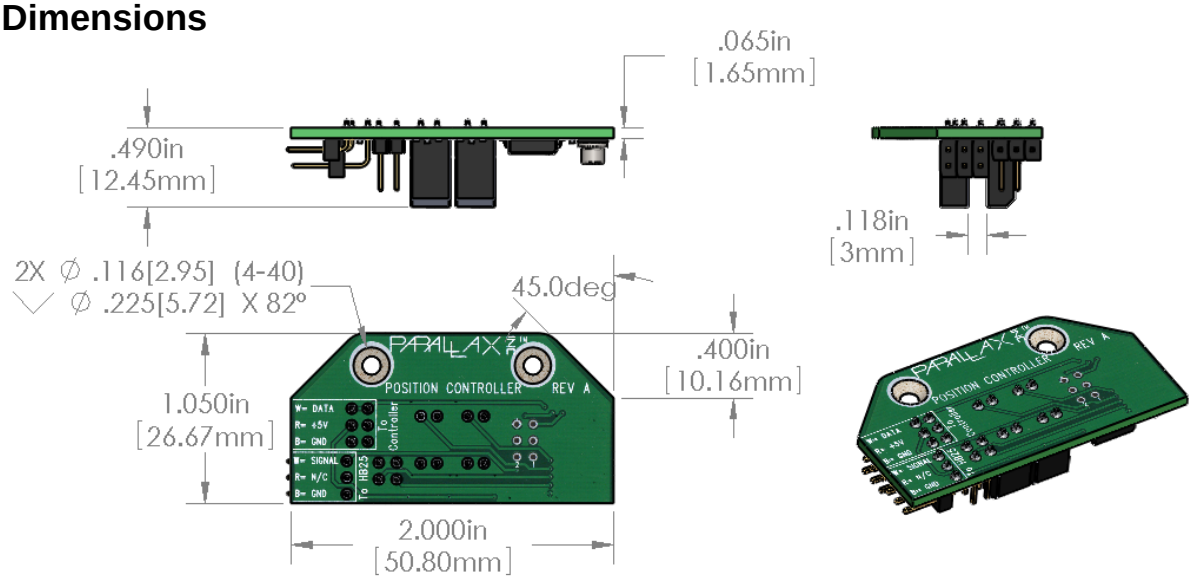


Figure 4: *Example of data exchange showing the QSPD command (0x10) addressed to ID Value 2 (0x02) and a reply of 30 positions/0.5 second (0x001E). Keep in mind that bytes are sent Least Significant Bit (LSB) First.*

Module Schematic



Dimensions



Example Code

Position Controllers are compatible with any microcontroller by following the previously stated communications protocol and instruction set. While there are many acceptable ways to execute commands in the instruction set, we have provided sample code for convenient implementation using PBASIC on the Basic Stamp 2. The code is for a two-wheeled robot with the right and left Position Controllers physically set to ID value 1 and 2 respectively. This code is available for download from the 27971 product page at www.parallax.com.

A.2. HB-25

Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

HB-25 Motor Controller (#29144)

General Description

The HB-25 motor controller combines the power of an H-bridge with the simplicity of a servo. The HB-25 is more than just a motor driver chip connected to some logic and high current on a PCB. A quality H-bridge involves an efficient thermal design, which is what the HB-25 has accomplished through a machined heat sink and fan to draw air over the H-bridge. Additionally, it uses a thermal bonding agent to transfer heat from the motor driver chip to the heat sink. The result is a high-current motor controller with great thermal characteristics and requiring no additional hardware for cooling.

Before using your HB-25 Motor Controller, read and understand this entire document, including the Precautions section beginning on page 4.

Features

- 25 A continuous current, 35 A surge @ 13.8 VDC
- Works with any size motor up to ½ HP
- Control a DC motor just like a continuous rotation servo
- A single pulse required to set motor speed
- Compatible with all Parallax microcontrollers
- 2 operation modes: control 1 or 2 HB-25's independently from a single I/O line
- Built-in automatic shut-off if invalid pulse widths are received
- Communication Timeout mode option for automatic shutoff

Application Ideas

- Robotics
- Automotive Applications

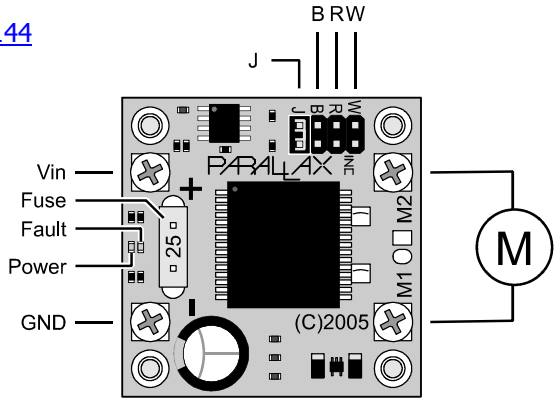
Resources and Downloads

Check out the HB-25 Motor Controller product page for the example source code and other resources:

http://www.parallax.com/detail.asp?product_id=29144

Interface Connections

Vin	Power input terminal, +6 to +16 VDC
GND	Negative side of battery terminal
M1 & M2	Motor connectors (Polarity reverses direction)
W	Servo pulse input
R	Not connected
B	Servo ground
J	Mode jumper pins



Modes of Operation

The HB-25 connects to the microcontroller much like a servo. You can use an extension cable (such as Parallax part #805-00012) or a custom cable to connect the HB-25 to your controller. Reversing the M1 and M2 connections to the motor effectively reverses the direction. The HB-25 has two modes of operation which are selected by the jumper labeled "J".

Mode 1 is Single Mode, where only one HB-25 is present on the microcontroller I/O line. In this mode, the HB-25 can be controlled with as little as a single pulse from your microcontroller.

Mode 2 is dual-mode, which is used when connecting a second HB-25 to a first HB-25, rather than connecting directly to the microcontroller. This powerful feature allows two HB-25's to be independently controlled through the same I/O line, saving I/O pins. Two HB-25's are required to use Mode 2.

Mode 1

Mode 1 is selected when the jumper labeled "J" is in place.

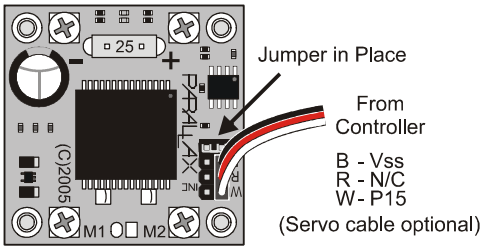


Figure 1: Single Mode Selection

Mode 1 Communication

In this mode, a single pulse value sent to the HB-25 can control the HB-25; no "refreshing" is necessary, as is the case with a servo. For compatibility however, you may send the HB-25 pulses every 20 ms just like a servo and it will function the same.

There is a hold-off time of 5 ms where the HB-25 will ignore incoming pulses. As a result, the unit should not be refreshed more frequently than about 5.25 ms + pulse time. Pulse time can be anywhere from 0.8 ms to 2.2 ms. If the HB-25 receives a pulse outside of this range, it will temporarily shut off the motor until it receives a valid pulse.

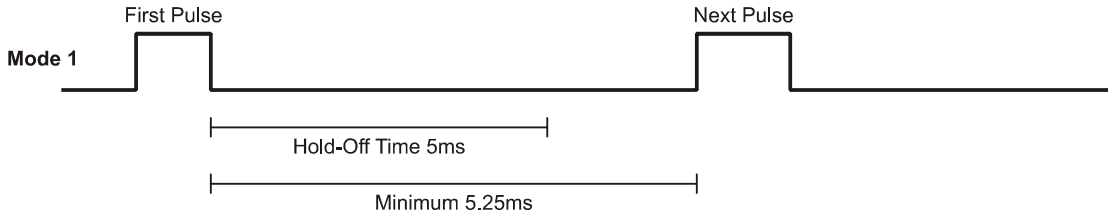


Figure 2: Single Mode Communication

The maximum time between pulses is unlimited, since a single pulse is all that is required to set the unit running indefinitely. This is especially nice for BASIC Stamp users with complex sensor code which may not have had sufficient time to refresh servos during the main loop. You only need to send a new pulse when you are ready to change the speed of the motor (or stop it).

A situation in which refresh pulses are necessary occurs when the Communication Timeout feature is enabled, as described on page 5.

Mode 2

Mode 2 is selected when the jumper is removed. Referring to Figure 3, Mode 2 should only be selected on Unit 2, which is the unit connected only to the other HB-25. Unit 1 needs to remain in Mode 1, with its jumper left in place.

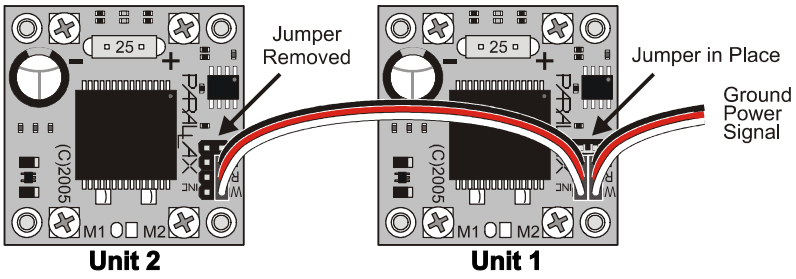


Figure 3: Dual Mode

The jumper is removed from Motor 2, and is in place in Motor 1

Mode 2 Communication

Independent control of both HB-25 units through a single I/O line is accomplished by sending two sequential pulses with a pause of 1.1 ms between them. When in Mode 2, the HB-25 looks for two sequential pulses and only responds to the second one. In reference to the diagram in Figure 4, Unit 1 would respond to Pulse #1, and Unit 2 would respond to Pulse #2. Unit 2 looks for Pulse #1, pauses for a 1 ms hold-off time, then looks for Pulse #2. Therefore, the timing between Pulse #1 and Pulse #2 is important, and there should be a minimum of 1.1 ms between pulses.

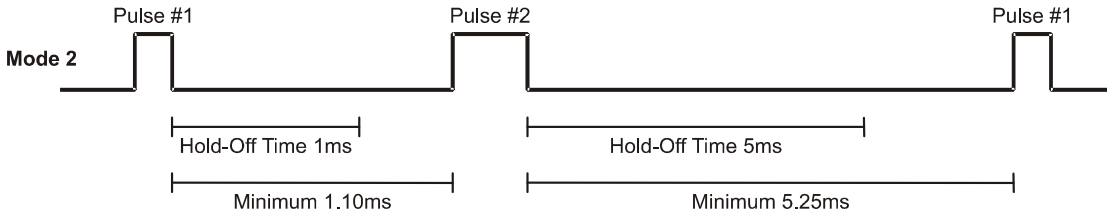


Figure 4: Dual Mode Communication

Just as in Mode 1, the HB-25 observes a 5 ms hold-ff time after the second pulse, so there should be a 5.25 pause after the second pulse. If a pulse were to start before the end of this hold-off time it might be seen as a shorter pulse than it is since the first part was cut off. This could cause a 2 ms pulse intended to be full speed in one direction to be interpreted as a shorter pulse, such as 1 ms, causing the motor to turn full speed in the opposite direction! Obviously this is not desirable, so observing the hold-off time is important to prevent this.

As a result, the unit should not be refreshed more frequently than about 5.25 ms + 1.1 ms + Pulse #1 + Pulse #2. Pulse time can be anywhere from 0.8 ms to 2.2 ms. If the HB-25 receives a pulse outside of this range, it will temporarily shut off the motor until it receives a valid pulse.

As with Mode 1, since each motor only requires a single pulse to set it indefinitely, there is no refreshing required unless the Communication Timeout feature is enabled.

BASIC Stamp 1 and 2 programmers using our example code will find that a PAUSE 1 command between Pulse #1 and Pulse #2 and a PAUSE 5 after Pulse #2 are adequate. This is because code execution time extends the resulting low times just enough to exceed the HB-25's own hold-off times. Those using

faster BS2 models or microcontrollers such as the SX will need to observe the minimum pause times more strictly.

Specifications

Motor Size	Any motor up to ½ HP max
Motor Supply	6.0 VDC – 16.0 VDC
Logic Supply	N/A – Internal Regulator
Load Current	25A Continuous 35A Surge (13.8 V)
Standby Current	50 mA @ 6 V, 80 mA @ 13.8 V (fan on)
PWM Frequency	9.2 kHz
Pulse Input	1.0 ms Full Reverse 1.5 ms Neutral (Off) 2.0 ms Full Forward
Pulse Refresh Rate	Not Required, Single Pulse Operation Capable
Modes	Single/Dual Motor Control
Protection Circuits	Over Voltage, Over Current, Over Temp
Automatic Fault Reset Indicators	Power (Green), Fault (Red)
Fuse	Mini ATC Standard 25 amp maximum
Cooling	Forced Air – Ball Bearing Fan
Terminals	Screw Posts with 35 A Rating
Weight	2.5 oz (71 grams)
Size	1.6" x 1.6" x 1.9"
Mounting	2 ea. 6-32 screws on .800" centers

Fuses

The HB-25 uses Mini ATC Standard fuses available from automotive supply sources. Always use the appropriate size fuse for the motor you are using. This can be determined by finding out the stall current of your motor. For example, your motor may draw only 5 to 10 amps during normal use but could draw 15 amps when stalled. In this case you would replace the 25 amp fuse with the 15 amp fuse. What this will do is protect your motor in case of a stall by blowing the fuse instead of the motor burning itself up.

Precautions

Testing for Reverse Polarity

There is *no reverse polarity protection* on the power input! If you are unsure of your wiring, before applying power do the following:

1. REMOVE THE FUSE from the HB-25. ***This is very important!***
2. BRIEFLY apply power to the unit.
3. If the Green Power LED did not illuminate, DISCONNECT POWER IMMEDIATELY as this indicates incorrect polarity; troubleshoot your wiring before continuing.
4. If the Green Power LED illuminated the polarity is correct.
5. Immediately remove power after verifying polarity. If the polarity is incorrect leaving power connected could overheat the power-sensing components.
6. REPLACE THE FUSE before continuing.

Safe Power-Up and Power-Down

Remember, because the HB-25 responds to pulses like a servo, changing states on your microcontroller's I/O pins at startup and shutdown can affect the HB-25 and set it into an unintended state. This includes the I/O pins changing between input and output as well as between output high and output low.

If your microcontroller is powered up or powered down while the HB-25 is powered up, the HB-25 could receive a false trigger. To prevent this, use one of the following options:

- Most reliable option:
    1. Power the microcontroller up first, using code that forces the program to wait until the HB-25 is powered up before sending pulses. Code examples are provided below.
    2. Power up the HB-25 for operation.
    3. When shutting down your application, cut off power to the HB-25 first.
    4. Power down your microcontroller last.
- Or, power the HB-25 and the microcontroller at the same time using a double-post (dual-circuit) switch.
- If there is the possibility that the HB-25 will be on before your microcontroller, use the example code as mentioned above to ensure that the HB-25 is properly initialized when the microcontroller is powered on. The I/O line connected to the HB-25 should be made LOW either before or immediately after the HB-25 powers up. If this does not happen the HB-25 may fail to respond to incoming pulses.

If your application poses any potential safety risk and you need the HB-25 to shut off the motors when the microcontroller has lost power or has malfunctioned, enable the Timeout Feature described below before operation.

Communication Timeout Mode

The HB-25 has a selectable Timeout mode. When this mode is enabled, the HB-25 will shut off the motor after 4 seconds if it does not receive pulses from the microcontroller. Once the motor is shut off, if the HB-25 receives a valid pulse again, it will restart the motor. Timeout mode should always be enabled in applications where a failure of your microcontroller could cause a dangerous situation or safety risk, or potentially cause damage to your application.

When this mode is enabled, the HB-25 will require refreshing by the microcontroller like a servo. These pulses could be 20 ms – 50 ms apart as would be typical with a servo, but may be up to 4 seconds apart.

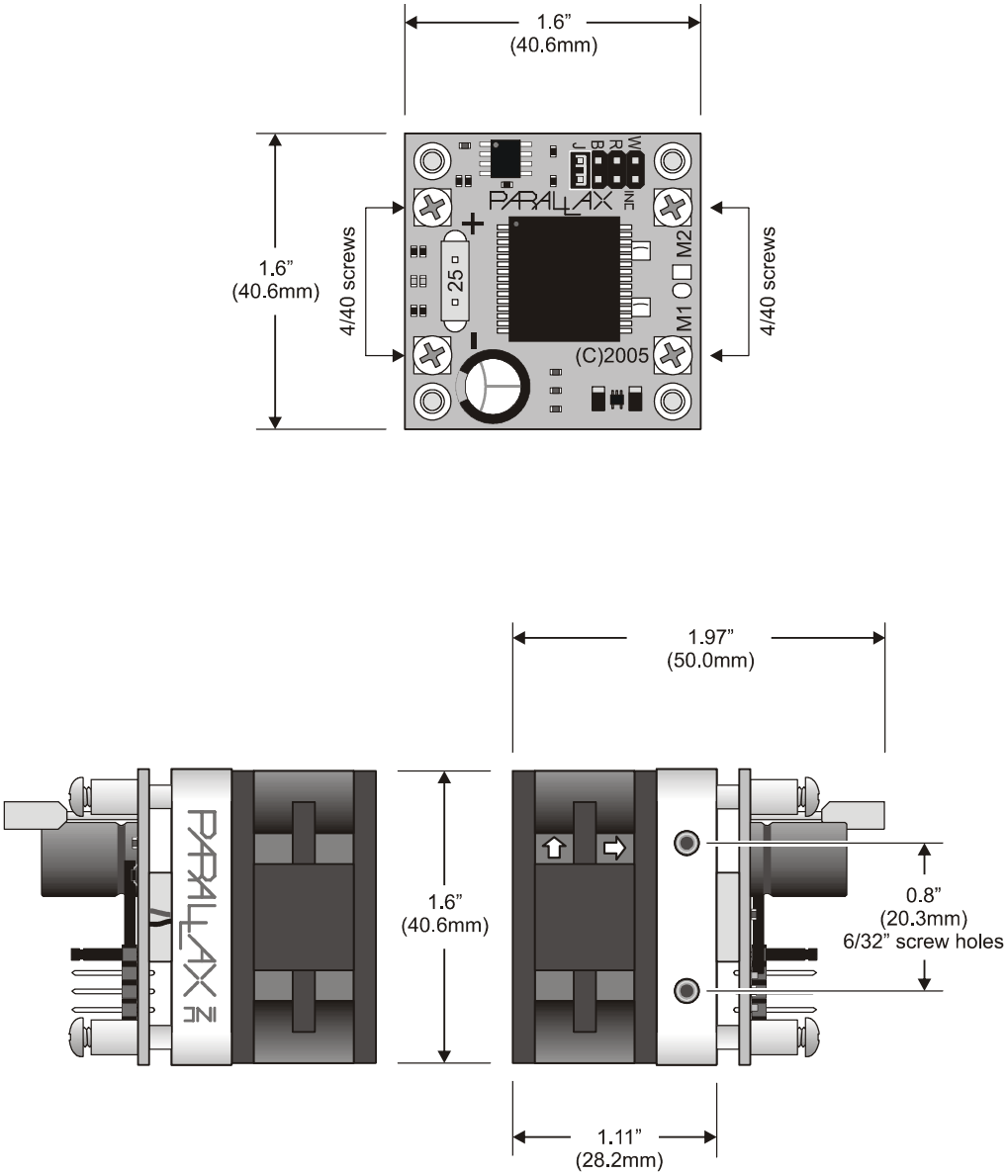
The Timeout mode can be toggled on and off by following the procedure below. The current state is stored in an on-board EEPROM and will remain until you toggle it again.

To toggle the state of the Timeout mode:

1. Disconnect the servo input cable from the HB-25. If there is a daisy-chained cable to a second unit remove it as well. It makes no difference whether the Mode Jumper (J) is installed.
2. Apply power to the HB-25.
3. With the power on change the state of the Mode Jumper by installing the jumper if it is removed or removing it if it is installed.
4. Remove and re-apply power.

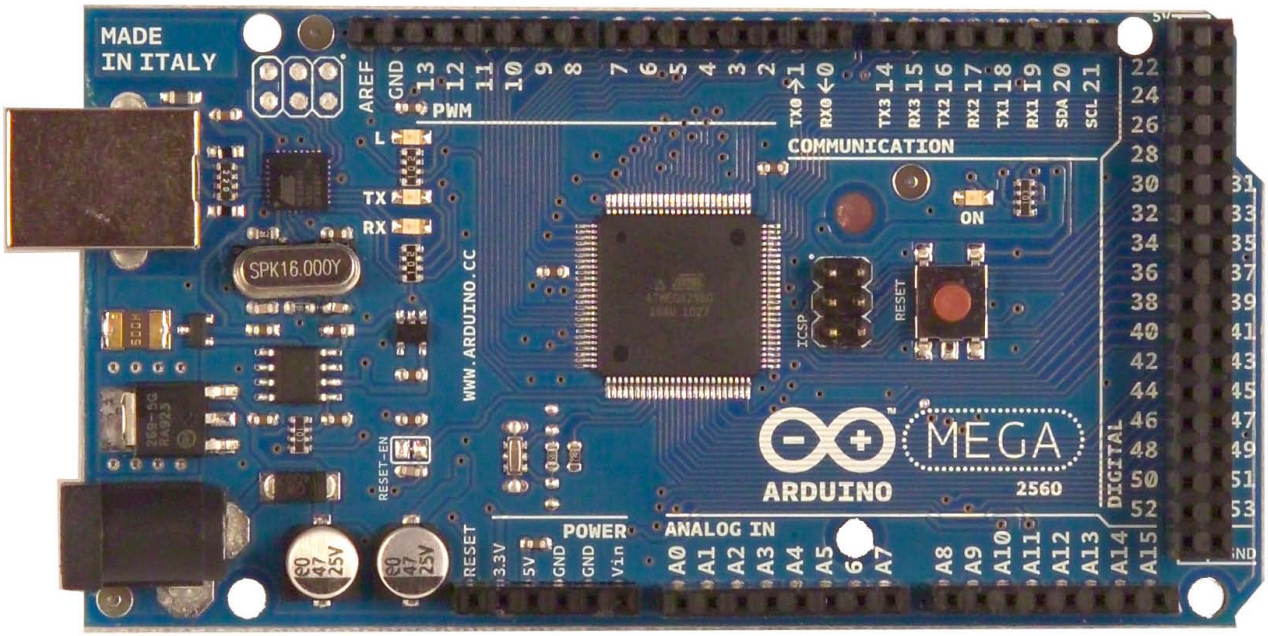
The timeout state should now be toggled. To change it back, repeat the above steps.

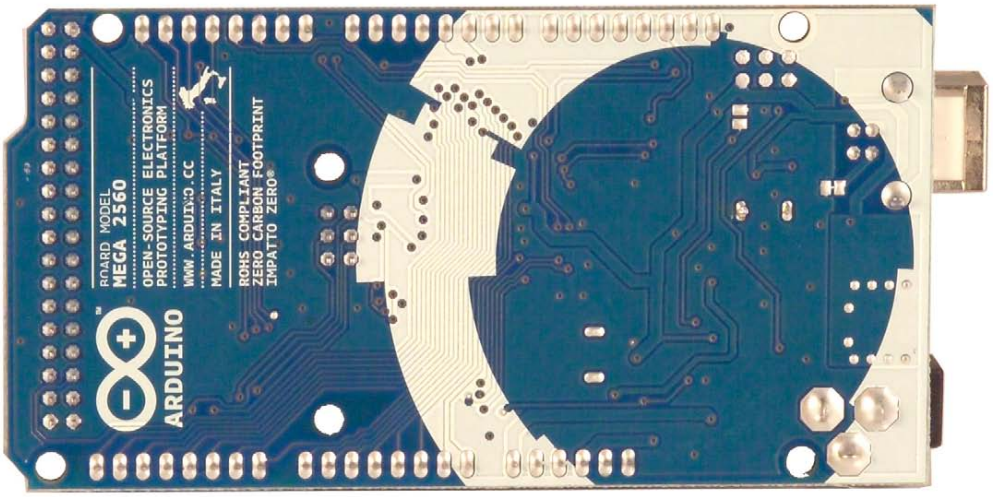
Module Dimensions



A.3. Arduino Mega 2560

Arduino Mega 2560





Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Schematic & Reference Design

EAGLE files: arduino-mega2560-reference-design.zip

Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.

- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins.

The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. For SPI communication, use the SPI library.

Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available in the Arduino repository. The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can

have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

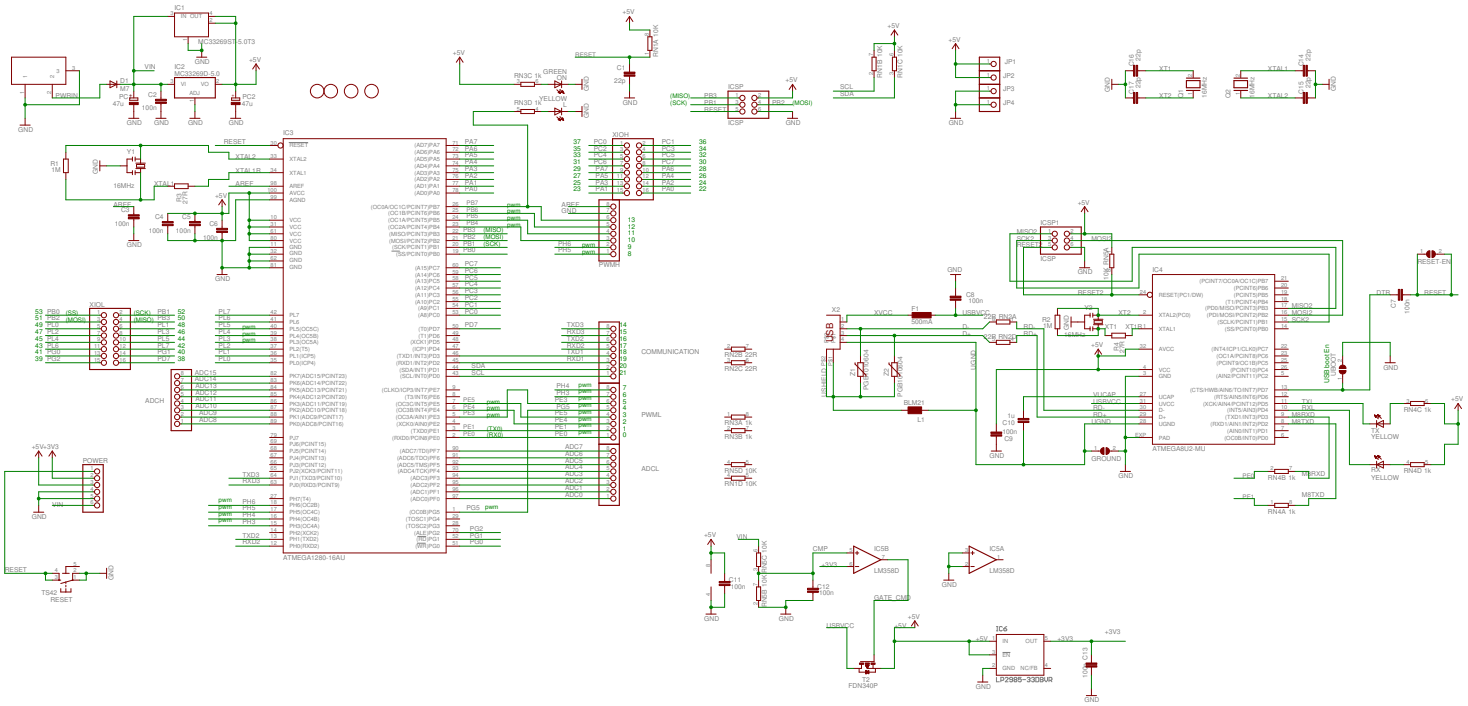
The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. *Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*

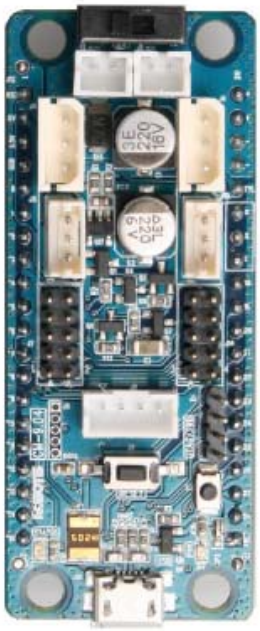
Arduino™ Mega 2560 Reference Design



A.4. Robotis OpenCM 9.04

1. OpenCM9.04 Hardware

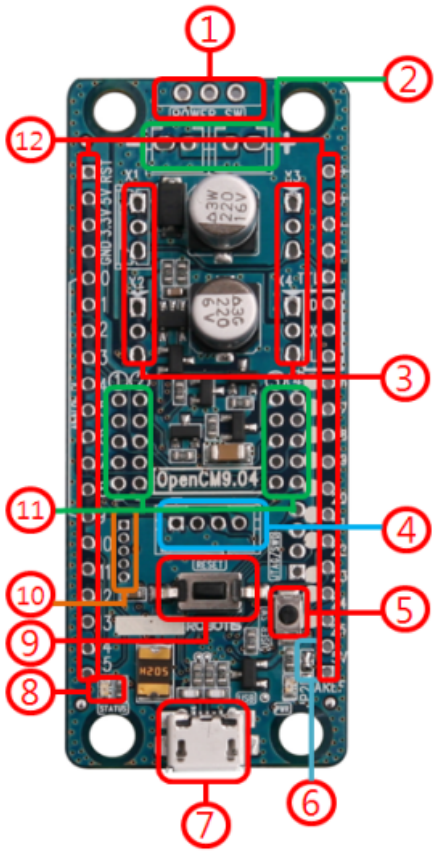
① Illustration of OpenCM9.04



[TOP]

[BOTTOM]

② Parts label



1. Power Switch: battery connection (power input).
2. Battery socket: sockets to connect Li-Ion Battery(ies). *[Note: each battery provides 3.7V; batteries in both pins provide 7.4V]*
3. DYNAMIXEL TTL BUS: ports for TTL-based Dynamixels. Connected dynamixels can be daisy-chained.
4. USART PIN: connect 4-pin devices such as the BT-110, BT-210, ZIG-110, LN-101 for UART communications (note: the LN-101 firmware only allows communications with the PC).
5. User Switch: user-programmable switch; OpenCM9.04 recovery-mode
6. Analog Reference Selection Jumper : change to 5V for analog signals.

7. Micro-B USB: Connect the OpenCM9.04 for communications, downloads, and 5V input power supply. Any type-B micro USB cable for Android phone is useful.

8. Status LED: Test LED for OpenCM9.04's programming. The LED blinks with a high/low signal to pin D16.

9. Reset switch: resets the CPU.

10. JTAG/SWD 4 PIN: Via JTAG/SWD terminal implement other programs such as IAR, Keil. The OpenCM9.04 has a total of 128Kbytes of memory, downloads and stores the bootloader's binary starting at 0x08000000. (Bootloader: 0x08000000 to 0x08002FFF. User programming space: 0x08003000 to 0x08020000).

11. Extenal Sensor PIN: pins for Robotis' sensors.

12. 2.54 mm GPIO Header: Interface external devices to the OpenCM9.04's STM32F103CB CPU.

③ Product package

A Type Board Only

package		quantity
controller	OpenCM9.04	1
manual	User Guide	1

B type

package		quantity
controller	OpenCM9.04	1
PIN Header	1x20 Pin Header	2
BOX Header	1x20 BOX Header	2
USB cable	Micro B Cable	1
manual	User Guide	1

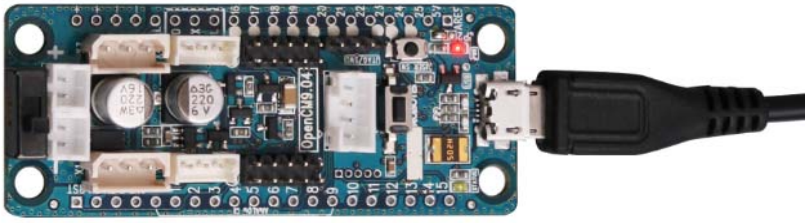
④ Product specifications

	OpenCM9.04
CPU	STM32F103CB (ARM Cortex-M3)
Op Voltage	5V~16V(USB 5V, DXL 12V, XL-Series 7.4V)
I/O	GPIO 26
Timer	8 (16bit)
Analog In(ADC)	10 (12bit)
Flash	128 Kbytes
SRAM	20 Kbytes
Clock	72Mhz(9 X 8 Mhz)
USB	1 (2.0 FullSpeed) Micro B type
CAN	1
USART	3
SPI	2
I2C(TWI)	2
Debug	JTAG & SWD
Extenal SenSor	4
3 Pin TTL	4(XL combo 3 PIN)
SW Tool	ROBOTIS OpenCM
SIZE	27mm X 66.5 mm

⑤ Power

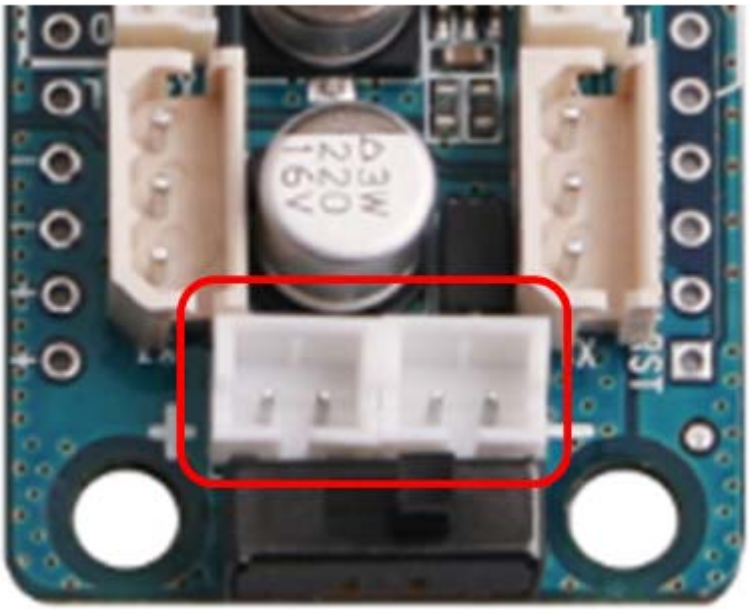
A. Connect to USB power

For programming simply connect the OpenCM9.04 via USB. The LED blinks I/O control under 5V becomes available.



B. Connect to battery power

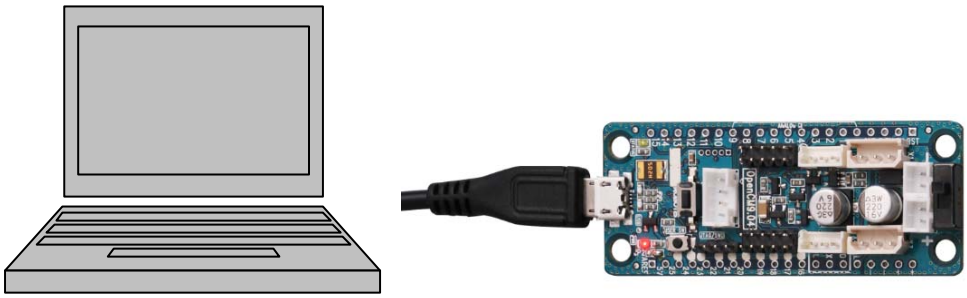
To control Dynamixel(s) connect the proper battery product. Afterwards the board runs the pre-downloaded mode.



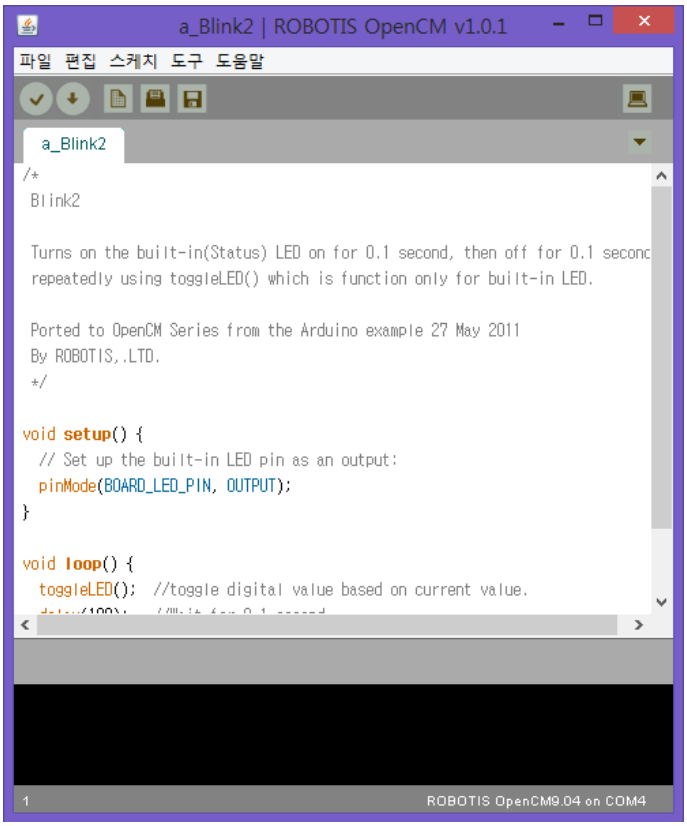
⑥ How-to-operate

A. The OpenCM9.04 executes the user code (0x08003000) when powered under battery or USB.

B. Connect to the PC via USB, write the program in the integrated development environment (IDE), compile and download the program.



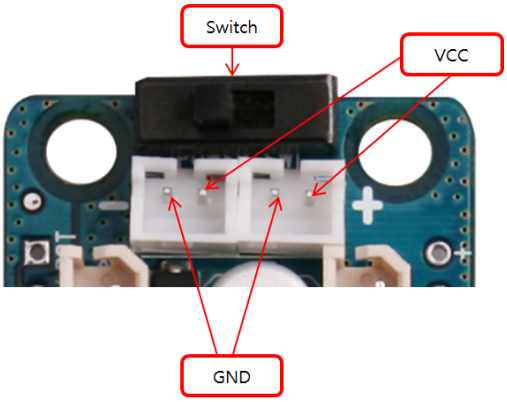
<connect the OpenCM9.04 to PC>



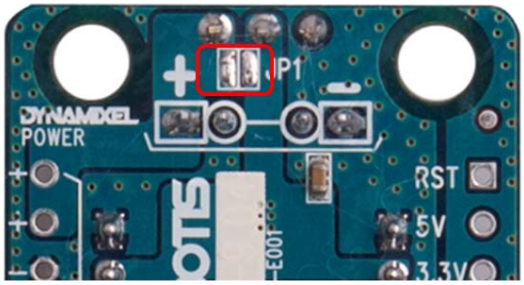
<OpenCM9.04 IDE ROBOTIS OpenCM>

⑦ Pin information

A. Power and switch

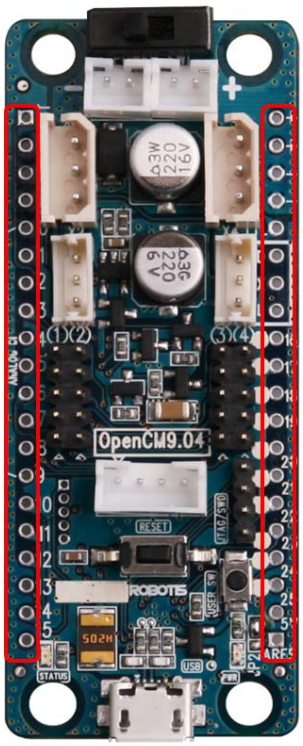


.

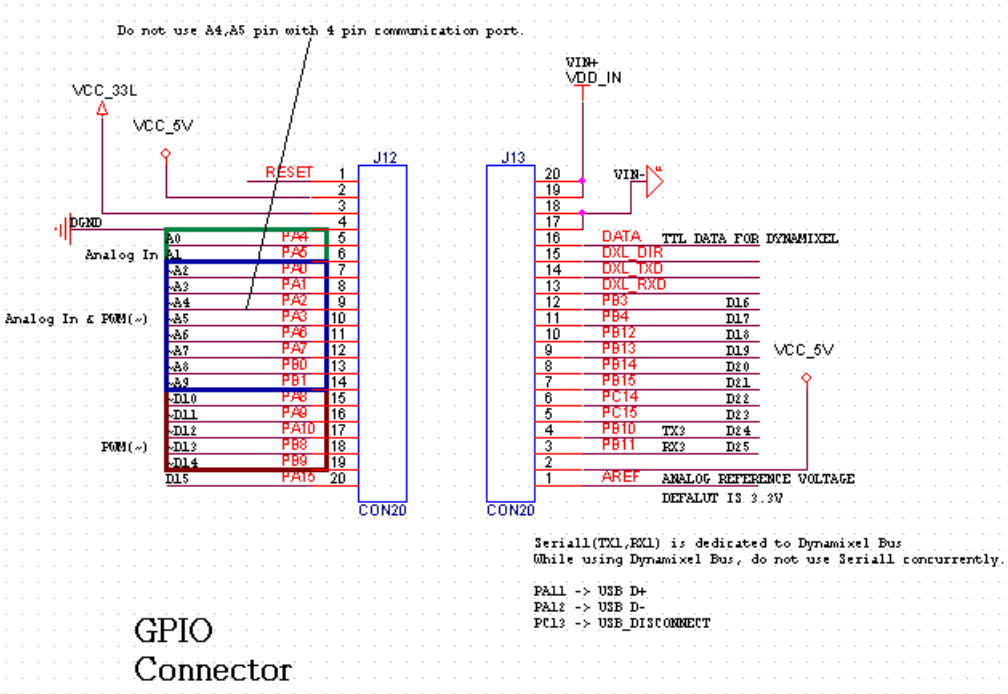


For switch use de-solder JP1

B. Refer to the OpenCM9.04's schematic for GPIO header pins to interface to the STM32F103CB CPU



[OpenCM9.04 GPIO]

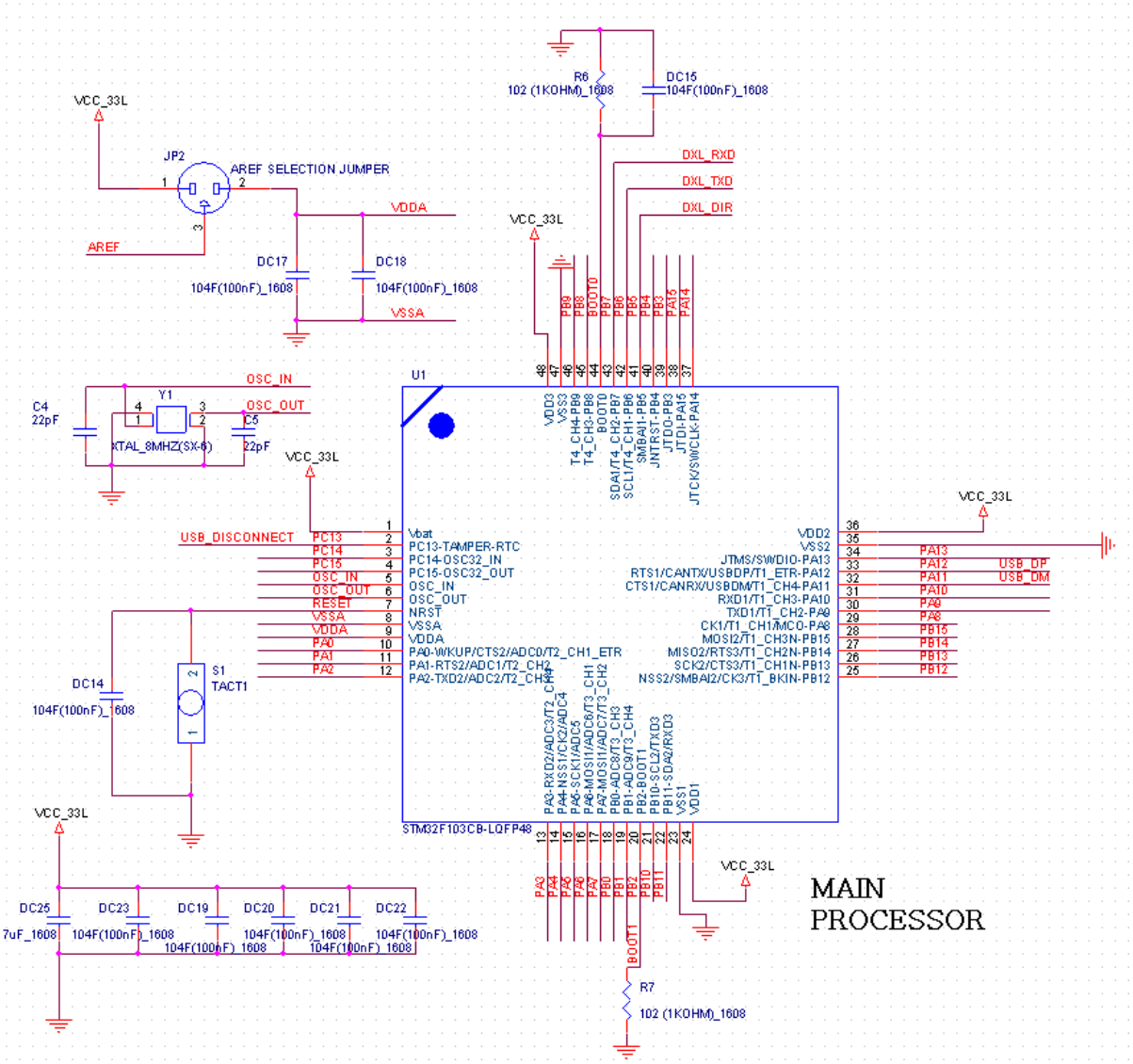


[OpenCM9.04 GPIO schematic]

For reference VDD is 12V; Dynamixel-related portions (pins PB5, PB6,

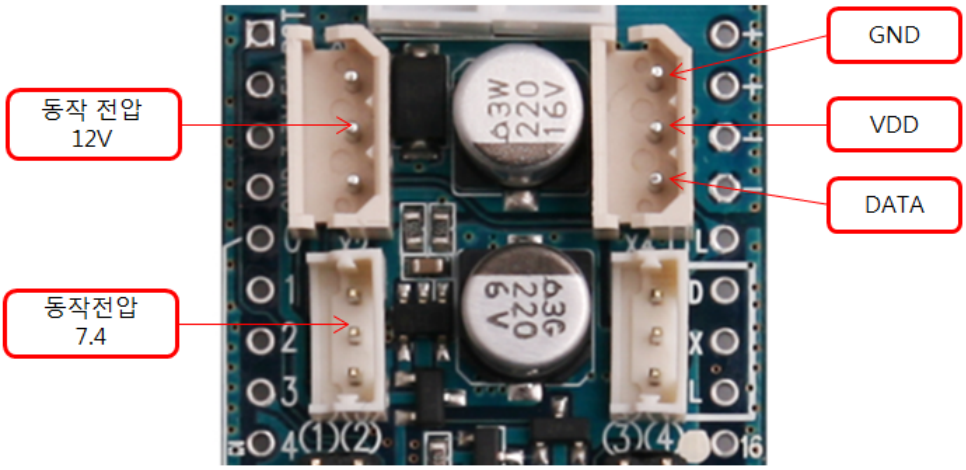
PB7) have been wired separately so they are not available for use

.

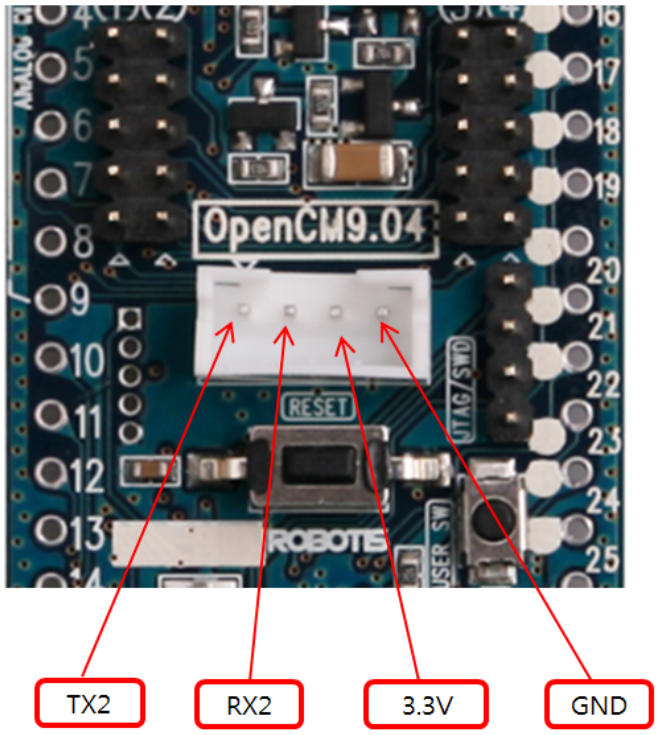


< STM32F103CB(LQFP48 Package) CPU connection schematic>

C. DYNAMIXEL TTL BUS



D. 4-pin communication devices



The same labels shown on the reverse side.

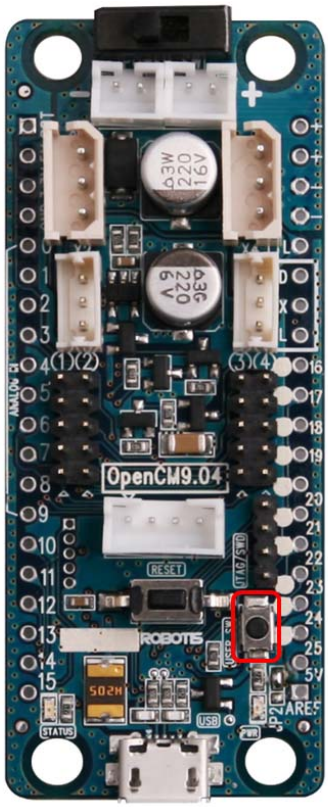
⑧ Schematic and PCB Gerber file (Schematic & Gerber Design)

The OpenCM9.04 resources (both hardware and software) are 100% open. Resources can be obtained via gitHub with the link below.

https://github.com/robotis-pandora/ROBOTIS-OpenCM.git

⑨ Emergency recovery mode

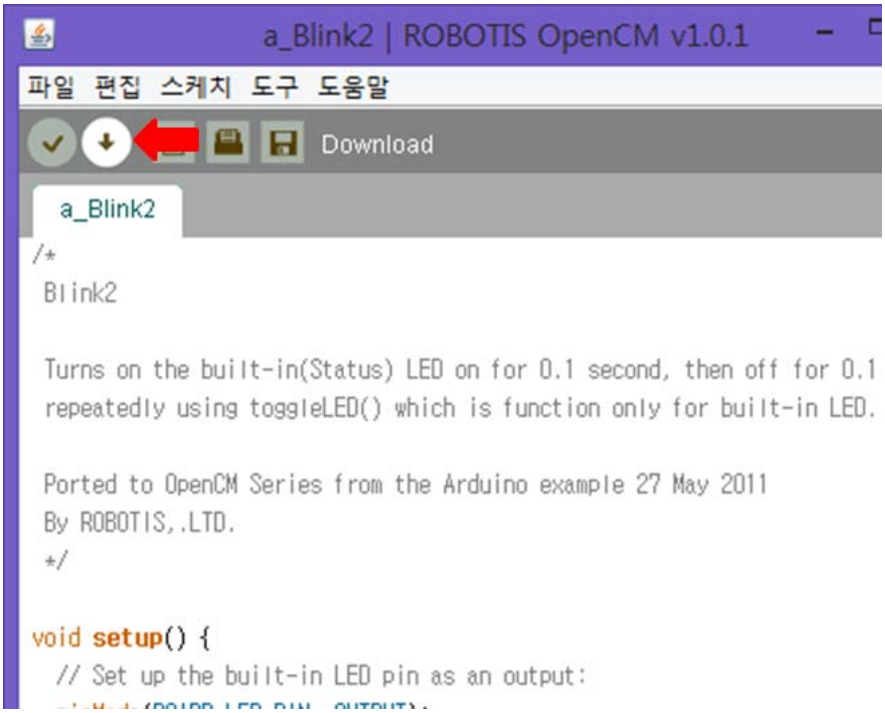
A. Whether the OpenCM9.04 USB drivers are not initialized nor device detected press the switch and connection will reestablish.



B. When connected via USB check the STATUS LED.

C. Go to File -> examples -> Digital -> Blink example; download it and

press the button (any other example OK).



A.5. Monitor AT070TN90

1. General Specifications

No.	Item	Specification	Remark
1	LCD size	7.0 inch(Diagonal)	
2	Driver element	a-Si TFT active matrix	
3	Resolution	800 × 3(RGB) × 480	
4	Display mode	Normally White, Transmissive	
5	Dot pitch	0.0642(W) × 0.1790(H) mm	
6	Active area	154.08(W) × 85.92(H) mm	
7	Module size	164.9(W) ×100.0(H) ×5.7(D) mm	Note 1
8	Surface treatment	Anti-Glare	
9	Color arrangement	RGB-stripe	
10	Interface	Digital	
11	Backlight power consumption	TBD	
12	Panel power consumption	TBD	
13	Weight	TBD	

Note 1: Refer to Mechanical Drawing.

2. Pin Assignment

TFT LCD Panel Driving Section

FPC Connector is used for the module electronics interface. The recommended model is FH12A-50S-0.5SH manufactured by Hirose.

Pin No.	Symbol	I/O	Function	Remark
1	V_{LED+}	P	Power for LED backlight (Anode)	
2	V_{LED+}	P	Power for LED backlight (Anode)	
3	V_{LED-}	P	Power for LED backlight (Cathode)	
4	V_{LED-}	P	Power for LED backlight (Cathode)	
5	GND	P	Power ground	
6	V_{COM}	I	Common voltage	
7	DV_{DD}	P	Power for Digital Circuit	
8	MODE	I	DE/SYNC mode select	Note 1
9	DE	I	Data Input Enable	
10	VS	I	Vertical Sync Input	
11	HS	I	Horizontal Sync Input	
12	B7	I	Blue data(MSB)	
13	B6	I	Blue data	
14	B5	I	Blue data	
15	B4	I	Blue data	
16	B3	I	Blue data	
17	B2	I	Blue data	
18	B1	I	Blue data	Note 2
19	B0	I	Blue data(LSB)	Note 2
20	G7	I	Green data(MSB)	
21	G6	I	Green data	
22	G5	I	Green data	
23	G4	I	Green data	
24	G3	I	Green data	
25	G2	I	Green data	

26	G1	I	Green data	Note 2
27	G0	I	Green data(LSB)	Note 2
28	R7	I	Red data(MSB)	
29	R6	I	Red data	
30	R5	I	Red data	
31	R4	I	Red data	
32	R3	I	Red data	
33	R2	I	Red data	
34	R1	I	Red data	Note 2
35	R0	I	Red data(LSB)	Note 2
36	GND	P	Power Ground	
37	DCLK	I	Sample clock	Note 3
38	GND	P	Power Ground	
39	L/R	I	Left / right selection	Note 4,5
40	U/D	I	Up/down selection	Note 4,5
41	V_{GH}	P	Gate ON Voltage	
42	V_{GL}	P	Gate OFF Voltage	
43	AV_{DD}	P	Power for Analog Circuit	
44	RESET	I	Global reset pin.	Note 6
45	NC	-	No connection	
46	V_{COM}	I	Common Voltage	
47	DITHB	I	Dithering function	Note 7
48	GND	P	Power Ground	
49	NC	-	No connection	
50	NC	-	No connection	

I: input, O: output, P: Power

Note 1: DE/SYNC mode select. Normally pull high.
When select DE mode, MODE=”1”, VS and HS must pull high.
When select SYNC mode, MODE= ”0”, DE must be grounded.

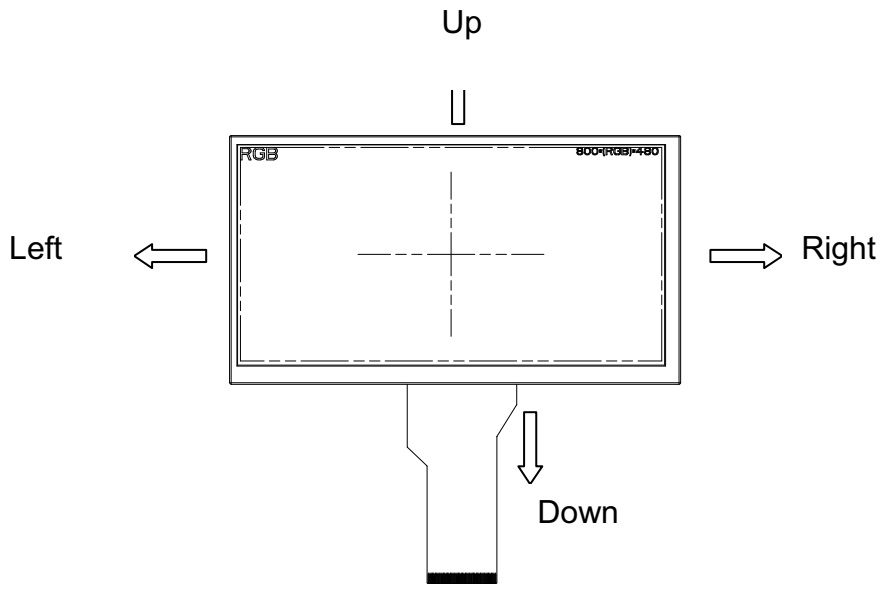
Note 2: When input 18 bits RGB data, the two low bits of R,G and B data must be grounded.

Note 3: Data shall be latched at the falling edge of DCLK.

Note 4: Selection of scanning mode

Setting of scan control input		Scanning direction
U/D	L/R	
GND	DV_{DD}	Up to down, left to right
DV_{DD}	GND	Down to up, right to left
GND	GND	Up to down, right to left
DV_{DD}	DV_{DD}	Down to up, left to right

Note 5: Definition of scanning direction.
Refer to the figure as below:



Note 6: Global reset pin. Active low to enter reset state. Suggest to connect with an RC reset circuit for stability. Normally pull high.

Note 7: Dithering function enable control, normally pull high.
When DITHB="1",Disable internal dithering function,
When DITHB="0",Enable internal dithering function,

3. Operation Specifications

3.1. Absolute Maximum Ratings

(Note 1)

Item	Symbol	Values		Unit	Remark
		Min.	Max.		
Power voltage	DV_{DD}	-0.3	5.0	V	
	AV_{DD}	6.5	13.5	V	
	V_{GH}	-0.3	40.0	V	
	V_{GL}	-20.0	0.3	V	
	V_{GH}-V_{GL}	-	40.0	V	
Operation Temperature	T_{OP}	-20	70	℃	
Storage Temperature	T_{ST}	-30	80	℃	
LED Reverse Voltage	V_R	-	1.2	V	Each LED Note 2
LED Forward Current	I_F	-	25	mA	Each LED

Note 1: The absolute maximum rating values of this product are not allowed to be exceeded at any times. Should a module be used with any of the absolute maximum ratings exceeded, the characteristics of the module may not be recovered, or in an extreme case, the module may be permanently destroyed.

Note 2: V_R Conditions: Zener Diode 20mA

3.1.1. Typical Operation Conditions

(Note 1)

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Power voltage	DV_{DD}	3.0	3.3	3.6	V	Note 2
	AV_{DD}	(10.2)	(10.4)	(10.6)	V	
	V_{GH}	(15.3)	(16.0)	(16.7)	V	
	V_{GL}	(-7.7)	(-7.0)	(-6.3)	V	
Input signal voltage	V_{COM}	-	TBD	-	V	
Input logic high voltage	V_{IH}	0.7 DV_{DD}	-	DV_{DD}	V	Note 3
Input logic low voltage	V_{IL}	0	-	0.3 DV_{DD}	V	

Note 1: Be sure to apply DV_{DD} and V_{GL} to the LCD first, and then apply V_{GH}.
Note 2: DV_{DD} setting should match the signals output voltage (refer to Note 3) of customer's system board.
Note 3: DCLK,HS,VS,RESET,U/D, L/R,DE,R0~R7,G0~G7,B0~B7,MODE,DITHB.

3.1.2. Current Consumption

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Current for Driver	I_{GH}	-	TBD	-	mA	V_{GH} =17.0V
	I_{GL}	-	TBD	-	mA	V_{GL} = -5.0V
	IDV_{DD}	-	TBD	-	mA	DV_{DD} =3.3V
	IAV_{DD}	-	TBD	-	mA	AV_{DD} =10.4V

3.1.3. Backlight Driving Conditions

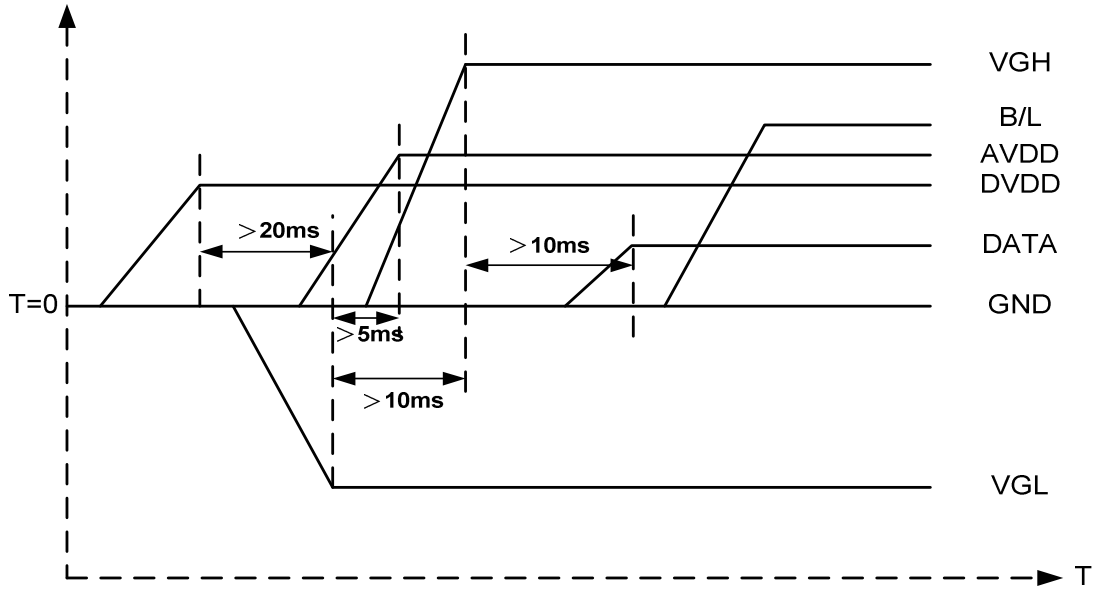
Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Voltage for LED backlight	V_L	(9.3)	(9.9)	(10.5)	V	Note 1
Current for LED backlight	I_L	(170)	(180)	(200)	mA	
LED life time	-	20,000	-	-	Hr	Note 2

Note 1: The LED Supply Voltage is defined by the number of LED at Ta=25℃ and I_L =180mA.

Note 2: The “LED life time” is defined as the module brightness decrease to 50% original brightness at Ta=25℃ and I_L =180mA. The LED lifetime could be decreased if operating I_L is lager than 180mA.

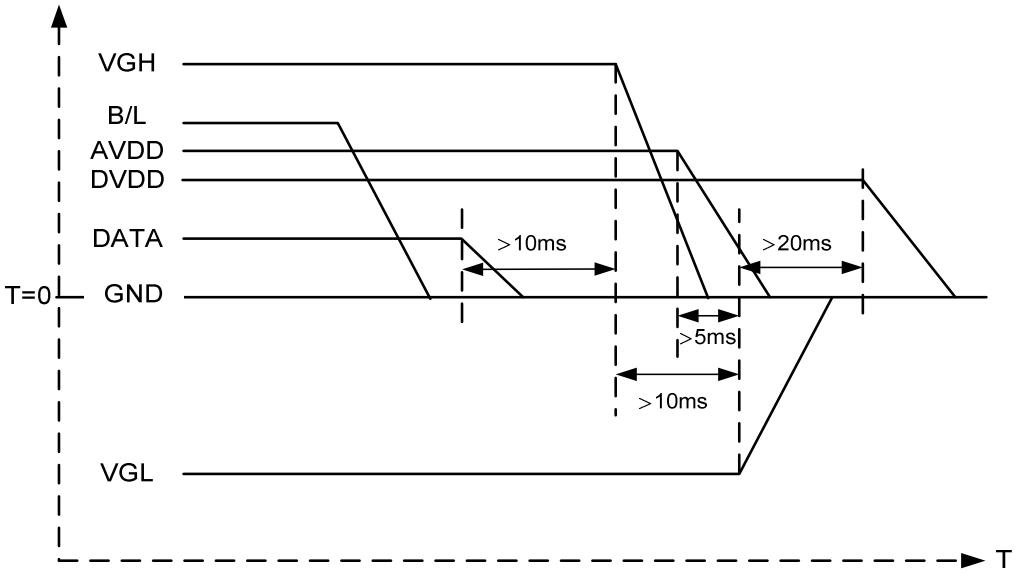
3.2. Power Sequence

a. Power on:



DV_{DD}→VGL→VGH→Data→B/L

b. Power off:



B/L→Data→VGH→VGL→DV_{DD}

Note: Data include R0~R7, B0~B7, GO~G7, U/D, L/R, DCLK, HS,VS,DE.

3.3. Timing Characteristics

3.3.1. AC Electrical Characteristics

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
HS setup time	T_{hst}	8	-	-	ns	
HS hold time	T_{hhd}	8	-	-	ns	
VS setup time	T_{vst}	8	-	-	ns	
VS hold time	T_{vhd}	8	-	-	ns	
Data setup time	T_{dsu}	8	-	-	ns	
Data hole time	T_{dhd}	8	-	-	ns	
DE setup time	T_{esu}	8	-	-	ns	
DE hole time	T_{ehd}	8	-	-	ns	
DV_{DD} Power On Slew rate	T_{POR}	-	-	20	ms	From 0 to 90% DV_{DD}
RESET pulse width	T_{Rst}	1	-	-	ms	
DCLK cycle time	T_{coh}	20	-	-	ns	
DCLK pulse duty	T_{cwh}	40	50	60	%	

3.3.2. Data Input Format

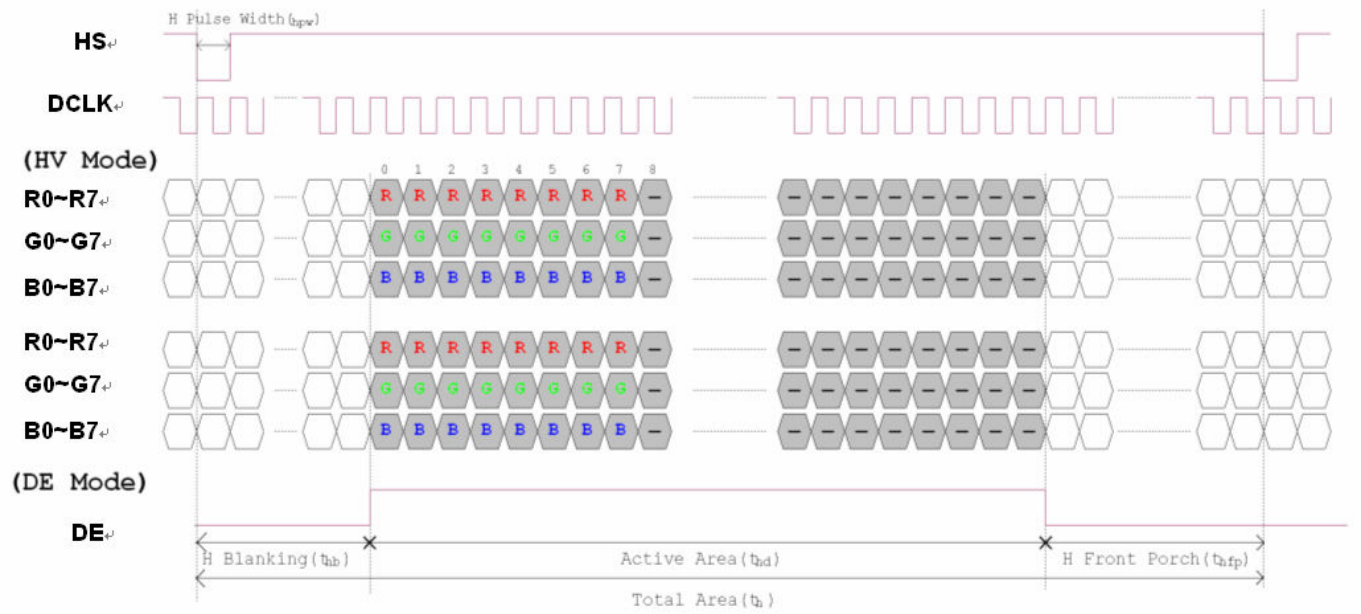


Figure 3. 1 Horizontal input timing diagram.

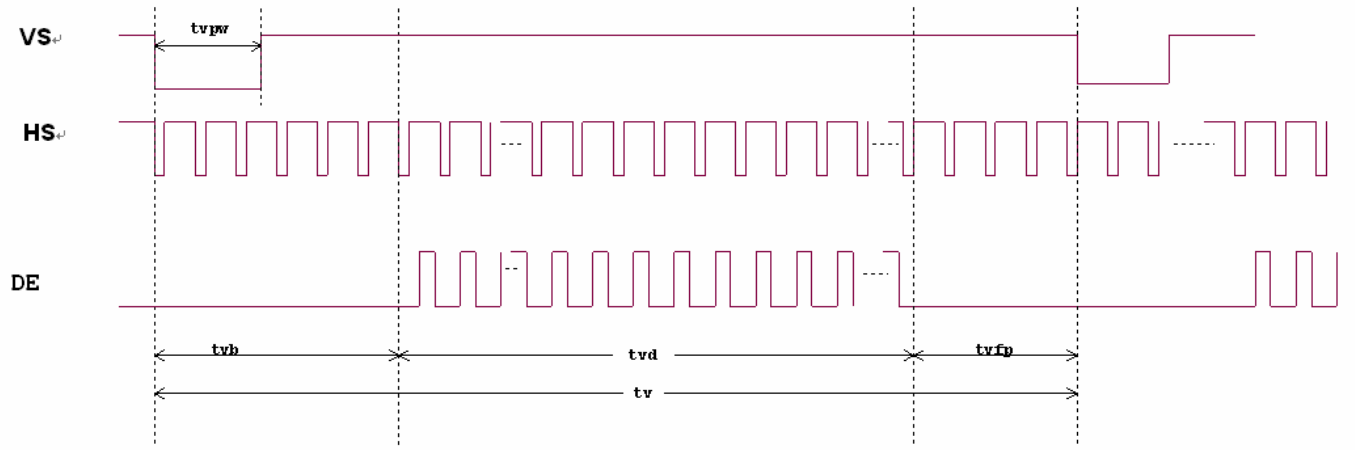


Figure 3. 2 Vertical input timing diagram.

3.3.3. Timing

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Horizontal Display Area	thd	-	800	-	DCLK	
DCLK Frequency	fclk	26.4	33.3	46.8	MHz	
One Horizontal Line	th	862	1056	1200	DCLK	
HS pulse width	thpw	1	-	40	DCLK	
HS Blanking	thb	46	46	46	DCLK	
HS Front Porch	thfp	16	210	354	DCLK	

Item	Symbol	Values			Unit	Remark
		Min.	Typ.	Max.		
Vertical Display Area	tvd	-	480	-	TH	
VS period time	tv	510	525	650	TH	
VS pulse width	tvpw	1	-	20	TH	
VS Blanking	tvb	23	23	23	TH	
VS Front Porch	tvfp	7	22	147	TH	

4. Optical Specifications

Item	Symbol	Condition	Values			Unit	Remark
			Min.	Typ.	Max.		
Viewing angle (CR≥ 10)	θ_L	Φ=180°(9 o'clock)	60	70	-	degree	Note 1
	θ_R	Φ=0°(3 o'clock)	60	70	-		
	θ_T	Φ=90°(12 o'clock)	40	50	-		
	θ_B	Φ=270°(6 o'clock)	60	70	-		
Response time	T_{ON}	Normal θ=Φ=0°	-	10	20	msec	Note 3
	T_{OFF}		-	15	30	msec	Note 3
Contrast ratio	CR		400	500	-	-	Note 4
Color chromaticity	W_X		0.26	0.31	0.36	-	Note 2 Note 5 Note 6
	W_Y		0.28	0.33	0.38	-	
Luminance	L		200	250	-	cd/m²	Note 6
Luminance uniformity	Y_U		70	75	-	%	Note 7

Test Conditions:

1. DV_{DD}=3.3V, I_L=180mA (Backlight current), the ambient temperature is 25℃.
2. The test systems refer to Note 2.

Note 1: Definition of viewing angle range

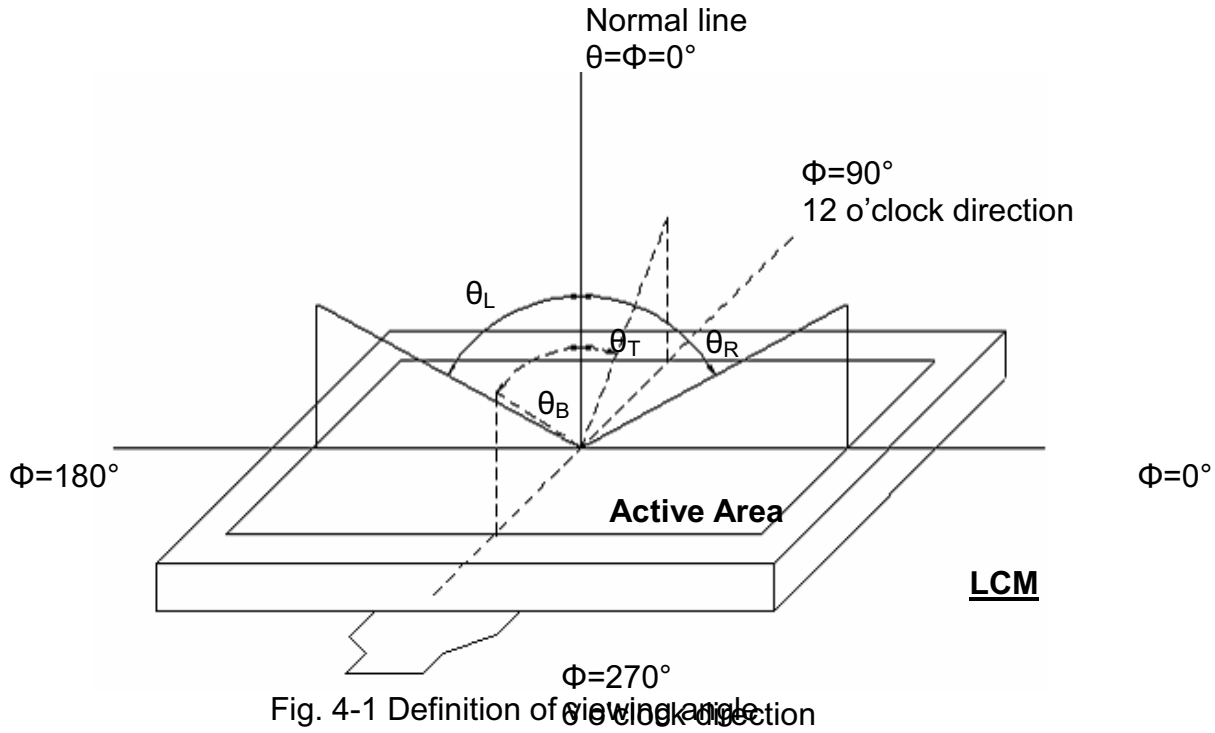


Fig. 4-1 Definition of viewing angle

Note 2: Definition of optical measurement system.

The optical characteristics should be measured in dark room. After 30 minutes operation, the optical properties are measured at the center point of the LCD screen. (Response time is measured by Photo detector TOPCON BM-7, other items are measured by BM-5A/Field of view: 1° /Height: 500mm.)

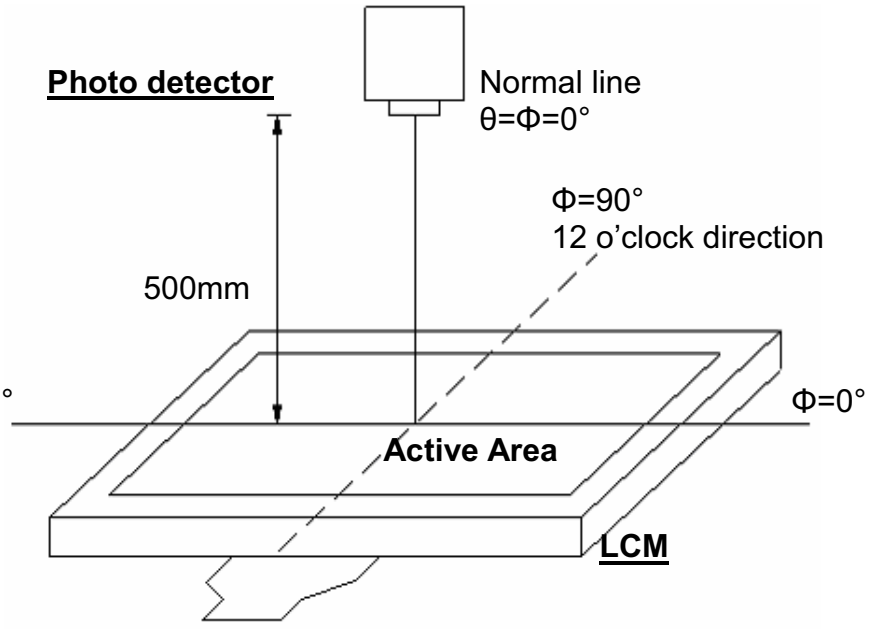


Fig. 4-2 Optical measurement system setup

Note 3: Definition of Response time

The response time is defined as the LCD optical switching time interval between “White” state and “Black” state. Rise time (T_{ON}) is the time between photo detector output intensity changed from 90% to 10%. And fall time (T_{OFF}) is the time between photo detector output intensity changed from 10% to 90%.

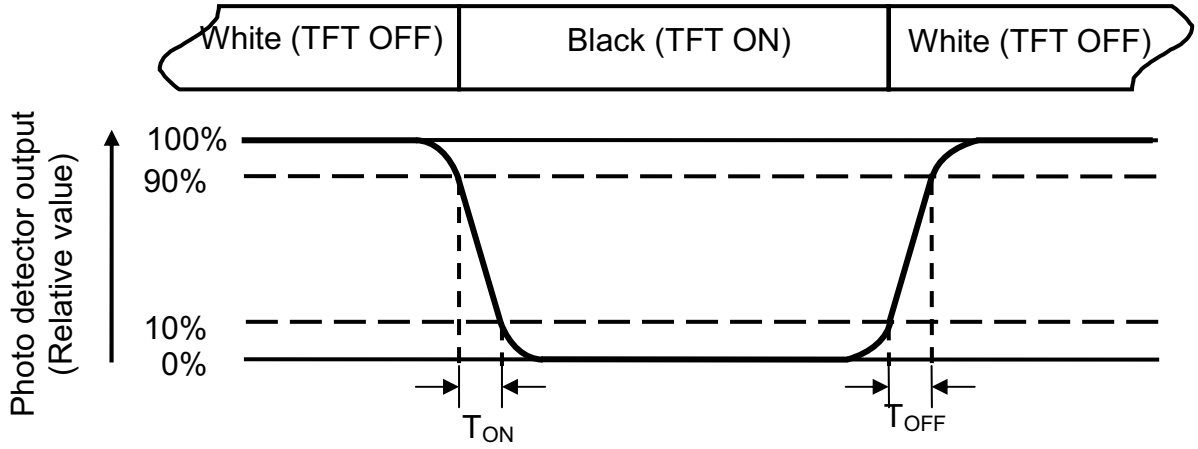


Fig. 4-3 Definition of response time

Note 4: Definition of contrast ratio

$$\text{Contrast ratio (CR)} = \frac{\text{Luminance measured when LCD on the "White" state}}{\text{Luminance measured when LCD on the "Black" state}}$$

Note 5: Definition of color chromaticity (CIE1931)

Color coordinates measured at center point of LCD.

Note 6: All input terminals LCD panel must be ground while measuring the center area of the panel. The LED driving condition is I_L=180mA .

Note 7: Definition of Luminance Uniformity

Active area is divided into 9 measuring areas (Refer to Fig. 4-4).Every measuring point is placed at the center of each measuring area.

$$Luminance\ \ Uniformity\ (Yu) = \frac{B_{min}}{B_{max}}$$

L-------Active area length W----- Active area width

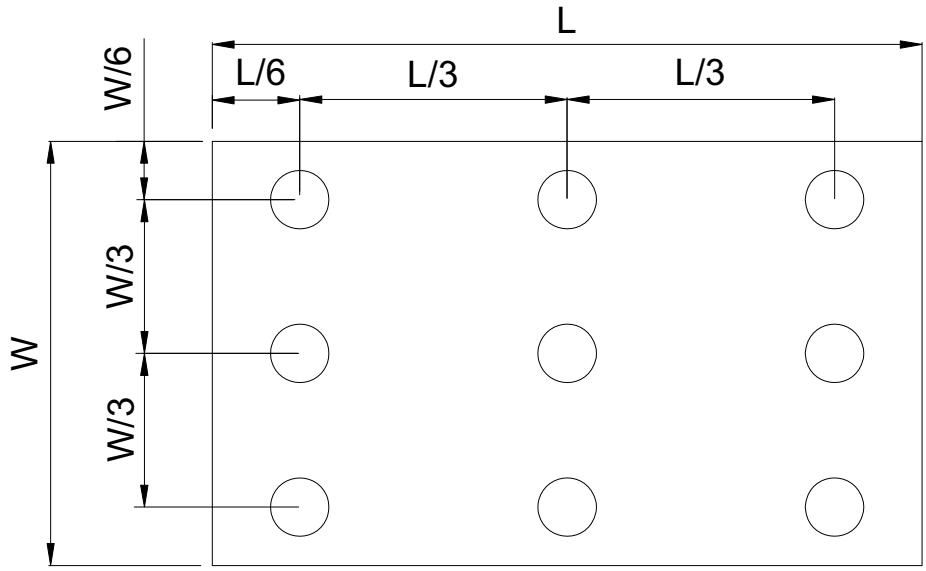


Fig. 4-4 Definition of measuring points

B_{max}: The measured maximum luminance of all measurement position.
B_{min}: The measured minimum luminance of all measurement position.

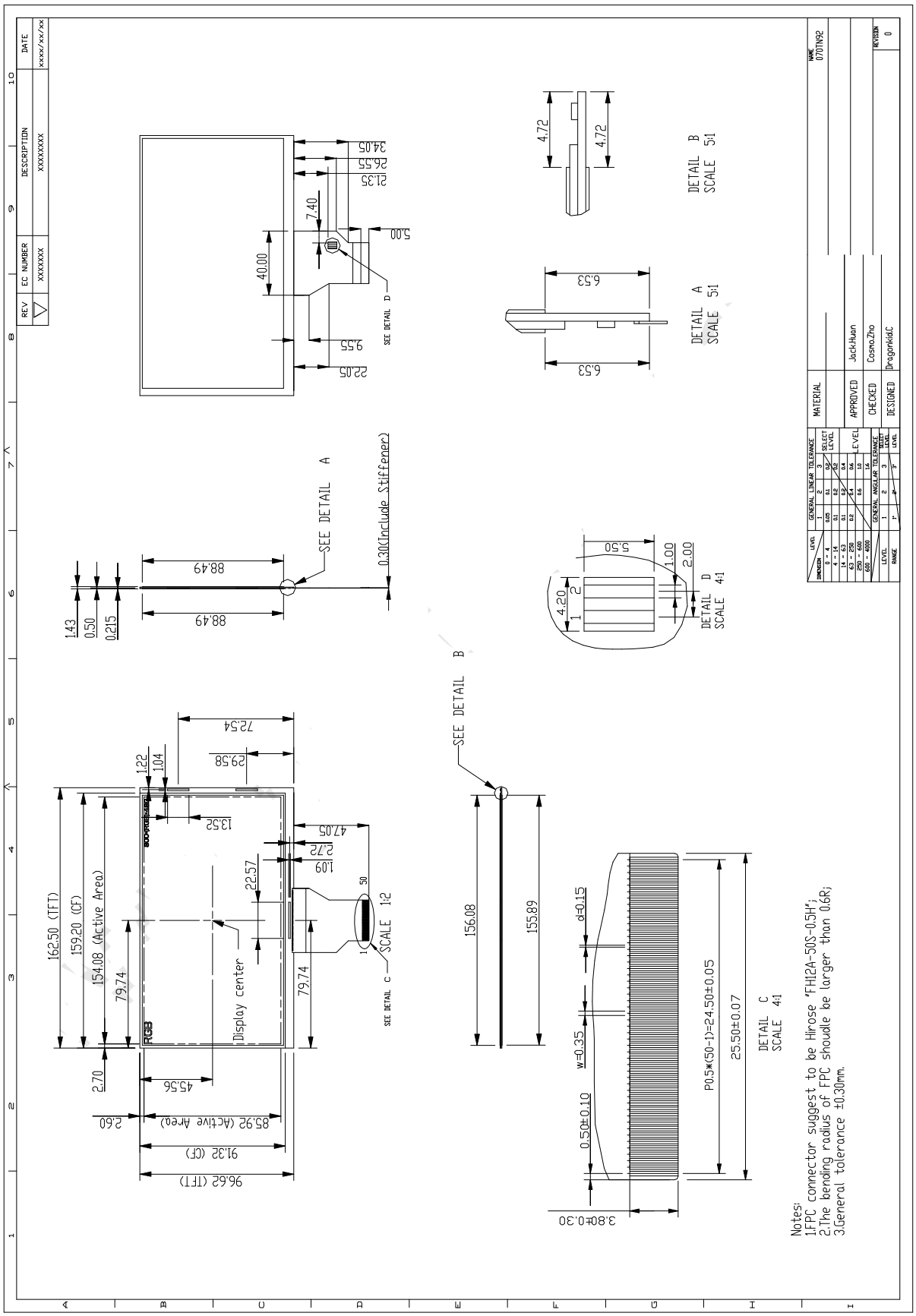
5. Reliability Test Items

(Note3)

Item	Test Conditions	Remark
High Temperature Storage	Ta = 80℃ 240hrs	Note 1，Note 4
Low Temperature Storage	Ta = -30℃ 240hrs	Note 1，Note 4
High Temperature Operation	Ts = 70℃ 240hrs	Note 2，Note 4
Low Temperature Operation	Ta = -20℃ 240hrs	Note 1，Note 4
Operate at High Temperature and Humidity	+60℃, 90%RH 240hrs	Note 4
Thermal Shock	-30℃/30 min ~ +80℃/30 min for a total 100 cycles, Start with cold temperature and end with high temperature.	Note 4
Vibration Test	Frequency range:10~55Hz Stroke:1.5mm Sweep:10Hz~55Hz~10Hz 2 hours for each direction of X. Y. Z. (6 hours for total)	
Mechanical Shock	100G 6ms,±X, ±Y, ±Z 3 times for each direction	
Package Vibration Test	Random Vibration : 0.015G*G/Hz from 5-200HZ, -6dB/Octave from 200-500HZ 2 hours for each direction of X. Y. Z. (6 hours for total)	
Package Drop Test	Height:60 cm 1 corner, 3 edges, 6 surfaces	
Electro Static Discharge	± 2KV, Human Body Mode, 100pF/1500Ω	

Note 1: Ta is the ambient temperature of samples.
Note 2: Ts is the temperature of panel's surface.
Note 3: In the standard condition, there shall be no practical problem that may affect the display function. After the reliability test, the product only guarantees operation, but don't guarantee all of the cosmetic specification.
Note 4: Before cosmetic and function test, the product must have enough recovery time, at least 2 hours at room temperature.

7. Mechanical Drawing



B Estudio de curvas de Bézier

Para definir las curvas de Bézier se definieron los puntos $\mathbf{P_0}$ y $\mathbf{P_3}$ como:

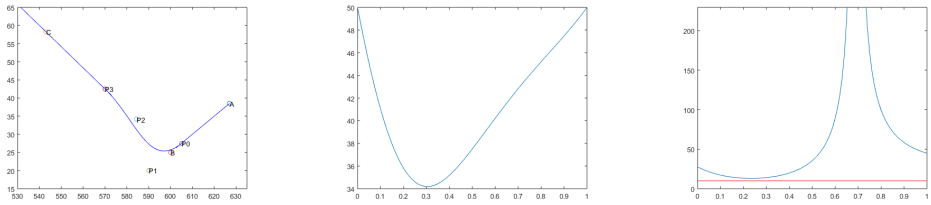
$$\begin{cases} \mathbf{P_{0_i}} = \mathbf{A} + (1-k)\overrightarrow{\mathbf{AB}} \\ \mathbf{P_{0_i}} = \mathbf{B} + k\overrightarrow{\mathbf{BC}} \\ k \in [0,0{,}5] \end{cases}$$

Siendo **A** el punto inicial del tramo recto anterior a la curva, **C** el punto final del tramo recto posterior y **B** el punto de intersección de ambas recta.

Para ver como cambia la curva al variar el parámetro k, se estudian cuatro casos diferentes donde varían principalmente las longitudes de los tramos y el ángulo que forman entre ellos. Se representa la curva en sí, y la evolución del módulo de la velocidad $\|\dot{\mathbf{B}}\|$ y del radio de curvatura R a lo largo de la curva. Con esto se tendrá una idea general sobre como varían estos parámetros para tener una visión cualitativa de la forma de la curva en función de k. Se establecen valores límite de velocidad máxima (1 m/s) y de radio de curvatura mínimo (10 cm).

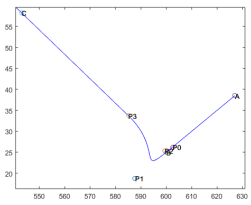
B.1. Caso 1

En primer caso de estudio, se cogen los siguientes puntos **A**, **B** y **C** que se ven en la Figura 1a. Se van variando los valores de k y se pueden ver los resultados en las figuras 1, 2 y 3.

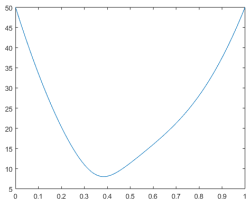


(a) Curva de Bézier. (b) Módulo de velocidad. (c) Radio de curvatura.

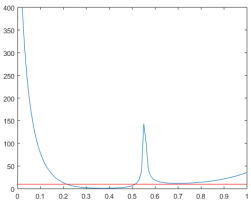
Figura 1: Gráficas de trayectoria, velocidad y curvatura con $k = 0{,}1$. Se puede ver como la velocidad no supera los valores máximos y como el radio de curvatura nunca es inferior al valor mínimo. La curva no hace giros bruscos ni da vueltas redundantes, teniendo una forma suave.



(a) Curva de Bézier.

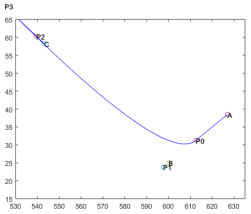


(b) Módulo de velocidad.

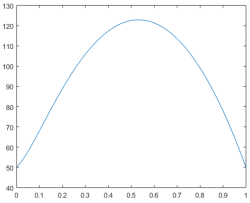


(c) Radio de curvatura.

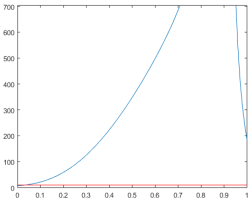
Figura 2: Gráficas de trayectoria, velocidad y curvatura con $k = 0,05$. Se puede ver como la velocidad es inferior al caso anterior, pero la curva no es tan suave. El radio de curvatura tiene un mínimo inferior al valor puesto como límite. El punto **B** prácticamente coincide con $\mathbf{P_0}$.



(a) Curva de Bézier.



(b) Módulo de velocidad.



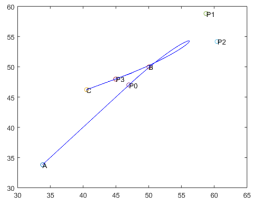
(c) Radio de curvatura.

Figura 3: Gráficas de trayectoria, velocidad y curvatura con $k = 0,25$. La curva en general es suave, pero tiene un giro ligeramente brusco al comienzo, cosa que se comprueba viendo el radio de curvatura al inicio (donde es inferior al valor límite durante un instante). El módulo de velocidad, sin embargo, excede el valor máximo.

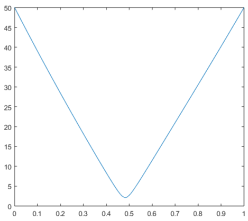
Se puede concluir que en esta situación, un valor de k muy bajo hace que la curva tenga una forma brusca y un radio de curvatura bajo, mientras que elevando el valor de k la curva se suaviza, pero aumenta el módulo de velocidad.

B.2. Caso 2

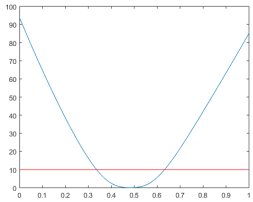
En este caso de estudio, se cogen los puntos **A**, **B** y **C** que se ven en la Figura 4a. Se ve como en este caso los puntos están más próximos entre sí, siendo la longitud de los tramos corta. Los resultados se ven en las figuras 4, 5, 6 y 7.



(a) Curva de Bézier.

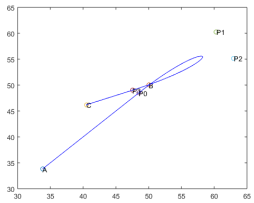


(b) Módulo de velocidad.

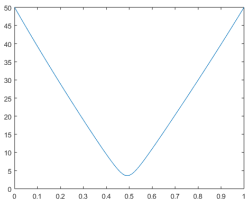


(c) Radio de curvatura.

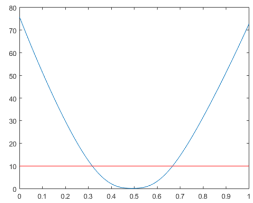
Figura 4: Gráficas de trayectoria, velocidad y curvatura con $k = 0,1$. La curva hace un giro que no es natural para conectar los tramos rectos. La velocidad no supera el valor límite, aunque tiene un mínimo muy bajo. El radio de curvatura también sale del límite establecido.



(a) Curva de Bézier.

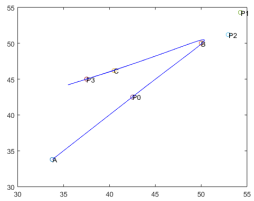


(b) Módulo de velocidad.

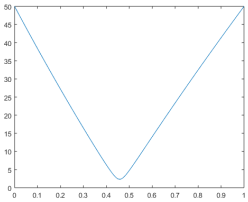


(c) Radio de curvatura.

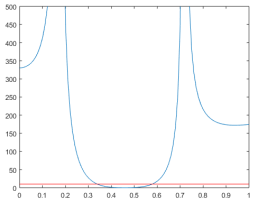
Figura 5: Gráficas de trayectoria, velocidad y curvatura con $k = 0,05$. Es un caso similar al de $k = 0,1$.



(a) Curva de Bézier.

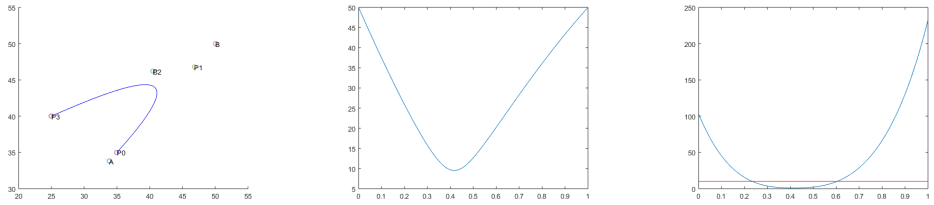


(b) Módulo de velocidad.



(c) Radio de curvatura.

Figura 6: Gráficas de trayectoria, velocidad y curvatura con $k = 0,25$. En este caso la curva es más natural que los casos anteriores, pero el radio de curvatura sigue pasando del límite establecido.



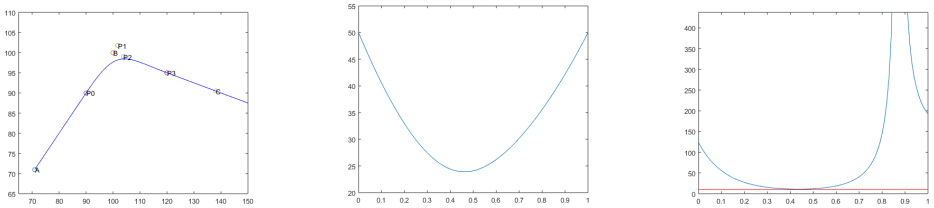
(a) Curva de Bézier. (b) Módulo de velocidad. (c) Radio de curvatura.

Figura 7: Gráficas de trayectoria, velocidad y curvatura con $k = 0,5$. . En este caso la curva tiene una forma suave y natural como la buscada. Sin embargo, el radio de curvatura sigue siendo inferior al valor límite.

Se puede concluir que en este caso, se necesitan valores más elevados de k que en B.1 para obtener una curva suave, pero aun así no se consigue una curva con el radio de curvatura buscado. Este hecho se debe a que los tramos rectos son de longitud corta y forman un ángulo más cerrado que en B.1 y por lo tanto la curva necesaria para unirlos ha de ser más cerrada.

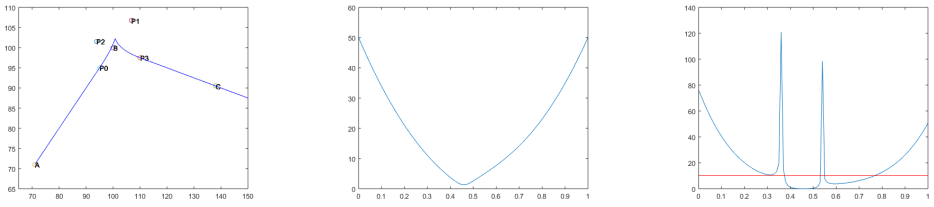
B.3. Caso 3

En este caso de estudio, se cogen los puntos **A**, **B** y **C** que se ven en la Figura 8a. Los resultados se ven en las figuras 8, 9 y 10.



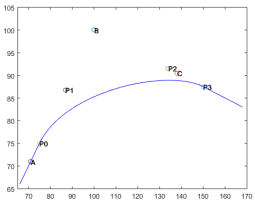
(a) Curva de Bézier. (b) Módulo de velocidad. (c) Radio de curvatura.

Figura 8: Gráficas de trayectoria, velocidad y curvatura con $k = 0,1$. La curva tiene una forma suave como la buscada, y tanto la velocidad como el radio de curvatura cumplen los requisitos buscados.

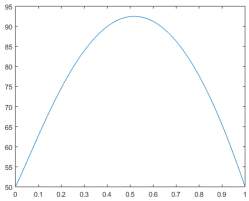


(a) Curva de Bézier. (b) Módulo de velocidad. (c) Radio de curvatura.

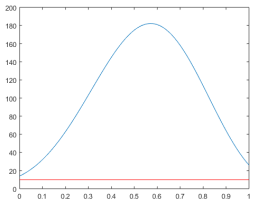
Figura 9: Gráficas de trayectoria, velocidad y curvatura con $k = 0,05$. La curva tiene una forma en punta con un giro muy brusco. La condición de curvatura no se cumple.



(a) Curva de Bézier.



(b) Módulo de velocidad.



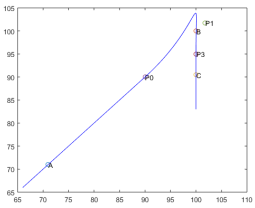
(c) Radio de curvatura.

Figura 10: Gráficas de trayectoria, velocidad y curvatura con $k = 0,25$. La curva tiene una forma muy suave, y se cumplen tanto las condiciones de velocidad como de curvatura. Sin embargo, la curva abarca una gran zona intermedia entre los dos tramos rectos, y es posible que entonces la curva quede fuera de la zona navegable del mapa.

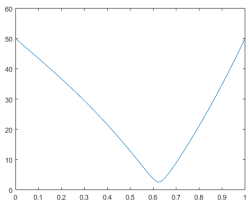
Se puede concluir de este caso que con tramos suficientemente largos, partiendo de valores de k bajos e incrementándolos, se consigue suavizar la curva y reducir su curvatura, aunque además de ir incrementando la velocidad, en casos que los tramos formen ángulos obtusos, es posible que la curva no sea navegable.

B.4. Caso 4

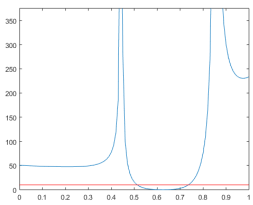
En este caso, se estudia una situación similar a la comentada en B.3, pero con tramos rectos formando un ángulo más cerrado. Los resultados se pueden ver en las figuras 11, 12 y 13.



(a) Curva de Bézier.

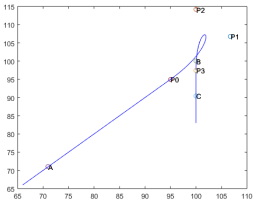


(b) Módulo de velocidad.

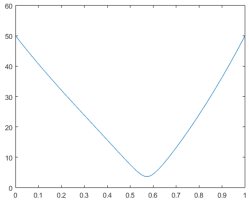


(c) Radio de curvatura.

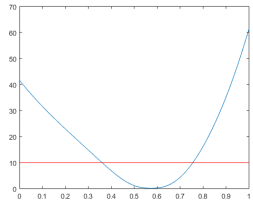
Figura 11: Gráficas de trayectoria, velocidad y curvatura con $k = 0,1$. . La curva es un poco cerrada, y no se cumple la condición de curvatura.



(a) Curva de Bézier.

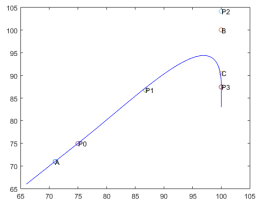


(b) Módulo de velocidad.

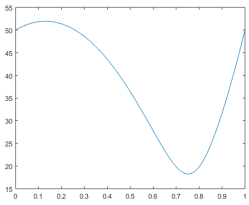


(c) Radio de curvatura.

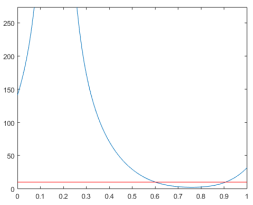
Figura 12: Gráficas de trayectoria, velocidad y curvatura con $k = 0,05$. . La curva hace un giro no natural y se incumple la condición de curvatura.



(a) Curva de Bézier.



(b) Módulo de velocidad.



(c) Radio de curvatura.

Figura 13: Gráficas de trayectoria, velocidad y curvatura con $k = 0,25$. . Como pasa en 10a, la curva es suave y se cumplen las condiciones de velocidad y curvatura, pero es posible que la curva no sea navegable.

De este estudio realizado, se puede concluir que no en todos los casos será posible obtener una curva como la deseada. El caso ideal es tener tramos largos y con ángulos obtusos, ya que de esta forma la curva se construye de forma más natural. Los valores de k bajos (próximos a cero) dan curvas bruscas pero con velocidades bajas; valores altos dan curvas más suaves y amplias, pero con valores de velocidad más elevados; además, una curva muy amplia puede quedar fuera de la zona navegable. Hay un compromiso con estas condiciones, pero normalmente un valor de k próximo a 0,1 da una curva como la buscada. En todo caso, el algoritmo de *path-finding* itera con varios valores de este parámetro hasta encontrar la primera curva que cumpla las condiciones de velocidad, curvatura y navegabilidad.

C Inicialización de MASHI

Updated: 2017/01/10

Mashi's setup

Software requirements:

Node.js: v.0.11.16

Instructions

Turn ON and connect the router to the web

Robot side computer :

- Before turning ON the computer, connect these devices:
  - Webcams
  - Microphone
  - Robotis and Arduino (USB Hub)
  - Wifi USB 5GHz adapter (optional)
  - Speaker
- Turn ON the computer
- Verify power options:
  - Dim the display: na
  - Turn off the display: never
  - Put the computer to sleep: never
- Verify audio settings (Playback and Recording devices. VERIFY DRIVERS)
  - Playback: Speakers / HP (Default device). Properties:
    - General: na
    - Levels:
      - Speakers (50%)
      - PC Speaker (off)
    - Enhancements
      - Bass Boost: no
      - Virtual Surround: no
      - Room Correction: no
      - Loudness Equalization: YES (settings: >50%)
  - Recording: Microphone 4-Microsoft LifeCam VX-3000 (Default device). Properties:
    - General: na
    - Listen:
      - Power management: Continue running when on battery power
    - Custom:
      - AGC? NO
    - Levels: 100%
    - Advanced:
      - Format: 1channel, 16bit, 16000Hz

- Exclusive mode:
    - Allow applications to take exclusive control of this device: YES ???
    - Give exclusive mode applications priority: YES ???
- o

- Connect to the Internet.
- Check robot IP with ipconfig, **IPROBOT**: e.g. 192.168.0.101
- FAILURE WITH USB connections (ARDUINO, WEBCAM, …). DON'T CONNECT TOO MUCH USB DEVICES TO THE SAME USB FISICAL PORT! USB HUB IS PREFERABLY TO CONNECT ARDUINO AND ROBOTICS CONNECTIONS. WEBCAM CAMERA USB CONNECT TO A DIFFERENT PORT! FINALLY WE HAVE PROBLEMS WITH THE USB HUB. MAYBE DON'T GIVE SUFFICIENT ENERGY TO THE USB DEVICES.
- Verify Arduino/robotis ports: start/devices and printers
    - Arduino: e.g. COM13
    - Robotis: e.g. COM19
- Go to workspace project path (ws): e.g. C:\Users\(user)\ws\Interface_original
- Modify serial ports in server.js if necessary
- In Command prompt:
    - Go to ws: cd C:\Users\NAO\ws\Interface_original
    - run: node server.js
- In command window verify the serial communication and movements (base and head) with keyboard:
    - Base
        - w: forward
        - s: backward
        - a: left
        - d: right
        - spacebar: stop base
    - Head:
        - j: zero position
        - u: pitch up
        - m: pitch down
        - h: yaw left
        - k: yaw right
        - y: roll left
        - i: roll right
- Open chrome browser: https://localhost:8000/robot.html
    - Set Text-to-speech
        - Voice language: e.g. Google Español
        - Volumen:
        - Rate
        - Pitch
        - Click Setear Configuración
- Connect display
    - Power source

- VGA/HDMI cable

Teleoperator side computer:

- Connect to Mashi LAN
- Connect microphone
- Connect headphones
- Open chrome browser: https://IPROBOT:8000/operator.html