



Guidelines for the control of a multiaxial planar robot with ST10F276

Introduction

This application note describes how to implement a PID control with the ST10F276 16-bit microcontroller for the control of a multiaxial planar robot.

The document provides guidelines for the complete development of a control system, able to fulfill all the requirements needed to drive an industrial manipulator.

The first chapter is an introduction to the robotic manipulators. It focuses on their working space, forward kinematics and the problem of the inverse kinematics. In particular it describes the main characteristics of an industrial wafer handler used as a case study for a multiaxial planar manipulator family.

The second chapter is a brief description of the ST10F276 16-bit microcontroller with a focus on its architecture and its peripherals. Moreover, an overview is given of the control board, named Starter Development Kit - ST10F276 and its three dedicated connectors for motion control.

The third chapter provides an overview of the hardware and mechanical equipment of a wafer handler. More specifically, it describes the encoder conditioning and motor driver circuits.

The fourth chapter is dedicated to the description of the basic routines for implementing PID control. The inverse kinematics of the wafer handler and the planning of the trajectory are also explained. The implementation of the *teach and repeat* technique and the *homing* procedure are shown.

See associated datasheets and technical literature for details of the components related to the devices and board used in this application note:

<http://www.st.com/stonline/books/ascii/docs/9944.htm> (L6205 Product Page)

Contents

1	An overview on robotics	3
1.1	Structure of a manipulator	3
1.1.1	Kinematics analysis	5
1.1.2	Singularity	6
1.2	The industrial wafer handler	6
1.2.1	Forward kinematics	7
1.2.2	Inverse kinematics	9
2	The SDK-ST10F276 control board	12
2.1	Brief description of the SDK-ST10F276	12
2.1.1	User Interfaces	13
2.1.2	On board motor control connectors	13
2.2	ST10F276 16-bit microcontroller - architectural overview	14
2.2.1	Basic CPU concepts	15
2.2.2	Memory organization	16
2.2.3	On-chip peripheral blocks	16
2.2.4	Managing Interrupts (hardware)	16
3	Hardware and mechanical equipments	17
3.1	The Dual DC motor and the power stage	17
3.2	Cables and connectors	20
3.3	The encoders and the conditioning circuit	20
3.4	Schematics of the driver board and the interface board.	23
4	Control algorithm	27
4.1	Motion and path planning	27
4.2	PID position control algorithm	32
4.3	Homing procedure	35
4.4	Teach and Repeat procedure	38
5	Revision history	40

1 An overview on robotics

1.1 Structure of a manipulator

A manipulator, from a mechanical point of view, can be seen as an open kinematic chain constituted of rigid bodies (links) connected in cascade by revolute or prismatic joints, which represent the degrees of mobility of the structure. These manipulators are also known as *serial manipulators*.

Only relatively few commercial robots are composed of a closed kinematic chain (parallel) structure. In this case there is a sequence of links that realize a loop. From this point on, we refer only to serial manipulators.

In the chain mentioned above, it is possible to identify two end-points: one end-point is referred to as the base, and it is normally fixed to ground, the other end-point of the chain is named the end-effector and is the functional part of the robot. The structure of an end effector, and the nature of the programming and hardware that drives it, depends on the intended task.

The overall motion of the structure is realized through a composition of elementary motions of each link respect to previous one.

A revolute joint allows a relative rotation about a single axis, and a prismatic joint permits a linear motion along a single axis, namely an extension or retraction.

It is assumed throughout that all joints have only a single degree-of-freedom: the angle of rotation in the case of a revolute joint, and the amount of linear displacement in the case of a prismatic joint.

The degrees of mobility must be suitably distributed along the mechanical structure in order to furnish the needed degrees of freedom (DOF) to execute a task. If there are more degrees of mobility than degrees of freedom the manipulator is said to be redundant.

The workspace of a point H of the end-effector is the set of all points which H occupies as the joint variables are varied through their entire ranges. The point H is usually chosen as either the center of the end-effector, or the tip of a finger, or the end of the manipulator itself. The workspace is also called work volume or work envelope.

Size and shape of the workspace depend on the coordinate geometry of the robot arm, and also on the number of degrees of freedom.

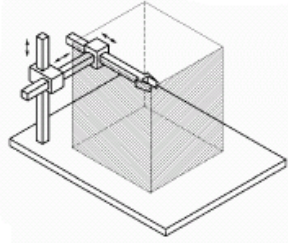
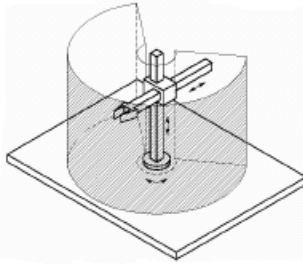
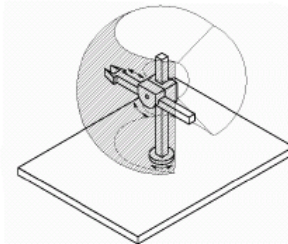
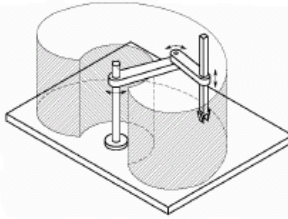
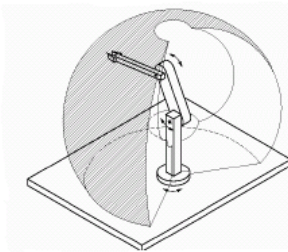
The workspace of a robot is a fundamental criterion in evaluating manipulator geometries. Manipulator workspace may be described in terms of the dexterous workspace and the accessible workspace. Dexterous workspace is the volume of space which the robot can reach with all orientations. That is, at each point in the dexterous workspace, the end-effector can be arbitrarily oriented.

The accessible workspace is the volume of space which the robot can reach in at least one orientation. In the dexterous workspace the robot has complete manipulative capability. However, in the accessible workspace, the manipulator's operational capacity is limited because of the terminal device can only be placed in a restricted range of orientations.

In other words, the dexterous workspace is a subset of the accessible workspace.

[Table 1](#) shows a classification of the manipulators accordingly to the type and sequence of the degrees of mobility of the structure, and of their workspaces.

Table 1. Open chain manipulators classification

Type	Workspace	Joints
Cartesian		<ul style="list-style-type: none"> – Three prismatic joints – A Cartesian degree of freedom corresponds to every joint
Cylindrical		<ul style="list-style-type: none"> – One revolute joint and two prismatic joints – Cylindrical coordinates
Spherical		<ul style="list-style-type: none"> – Two revolute joints and one prismatic joint – Spherical coordinates
SCARA		<ul style="list-style-type: none"> – Two revolute joints and one prismatic joint – Selective Compliance Assembly Robot Arm
Anthropomorphic		<ul style="list-style-type: none"> – Three revolute joints – It is the most dexterous structure

1.1.1 Kinematics analysis

Robot arm kinematics deals with the analytical study of the geometry of motion of a robot arm with respect to a fixed reference coordinate system as a function of time without regard to the forces/moments that causes the motion.

Thus, it deals with the analytical description of the spatial displacement of the robot as a function of time, in particular the relations between the joint space and the position and orientation of the end-effector of a robot arm.

In kinematics, we consider two issues:

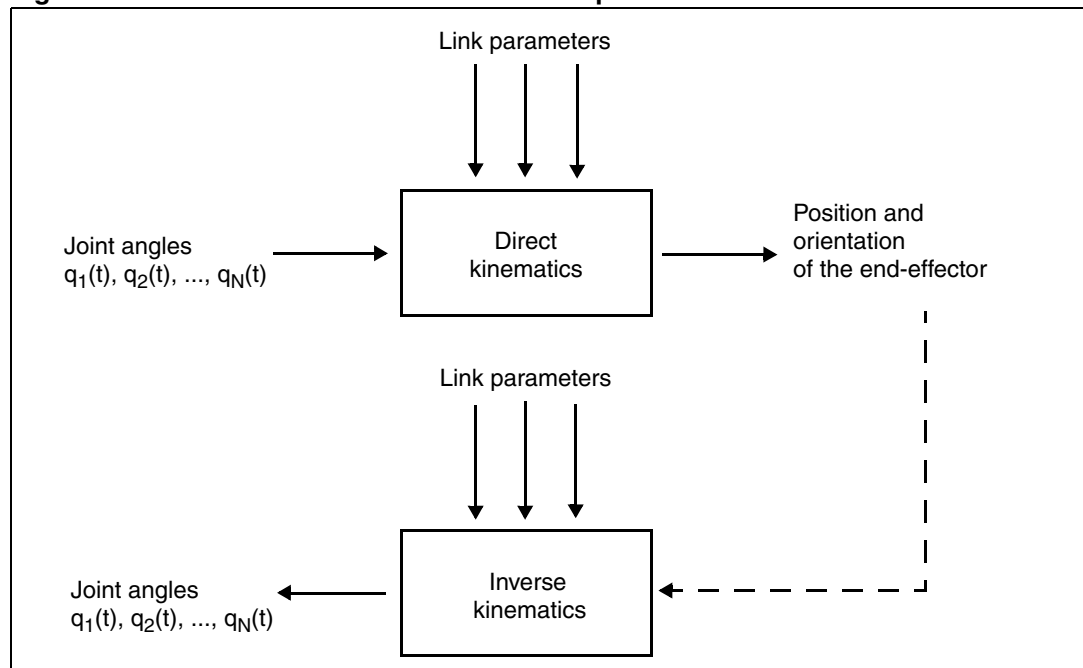
1. **Forward analysis:** for a given manipulator, given the joint angle vector $q(t) = (q_1(t), q_2(t), \dots, q_N(t))^T$ and the geometric link parameters, where n is the number of degrees of freedom, what is the position and orientation of the end-effector with respect to a reference coordinate system?
2. **Inverse analysis:** given a desired position and orientation of the end effector and the geometric link parameters with respect to a reference coordinate system, can the manipulator reach the desired manipulator hand position and orientation. And if it can, how many different manipulator configurations will satisfy the same condition?

For serial robots, the forward analysis problem is usually easy and straightforward. Unfortunately, the inverse analysis problem is of much more interest. For example, in industrial applications, the end-effector must follow some desired path; then, we need to find the joint angles for each position in the path.

For redundant robots, the inverse kinematics problem has then an infinite number of solutions. The extra degrees of freedom can then be used for other purposes, for example for fault tolerance, obstacle avoidance, or to optimize some performance criteria.

A simple block diagram indicating the relationship between these two problems is shown in the following figure.

Figure 1. The direct and inverse kinematics problems



Since the links of a robot arm may rotate and/or translate with respect to a reference coordinate frame, the total spatial displacement of the end-effector is due to the angular rotations and linear translations of the links.

Denavit and Hartenberg proposed a systematic and generalized approach of utilizing matrix algebra to describe and represent the spatial geometry of the links of a robot arm with respect to a fixed reference frame. This method uses a 4×4 homogeneous transformation matrix to describe the relationship between two adjacent rigid mechanical links and reduces the direct kinematics problem to finding an equivalent 4×4 homogeneous transformation matrix that relates the spatial displacement of the *hand coordinate frame* to the reference coordinate frame. These homogeneous transformation matrices are also useful in deriving the dynamic equation of motion of a robot arm.

In general, the inverse kinematics problem can be solved by several techniques. Most commonly used approaches are matrix algebraic, iterative, or geometric. A geometric approach based on both the link coordinate systems and the manipulator configuration has been used for the industrial wafer handler to which this application note refers.

1.1.2 Singularity

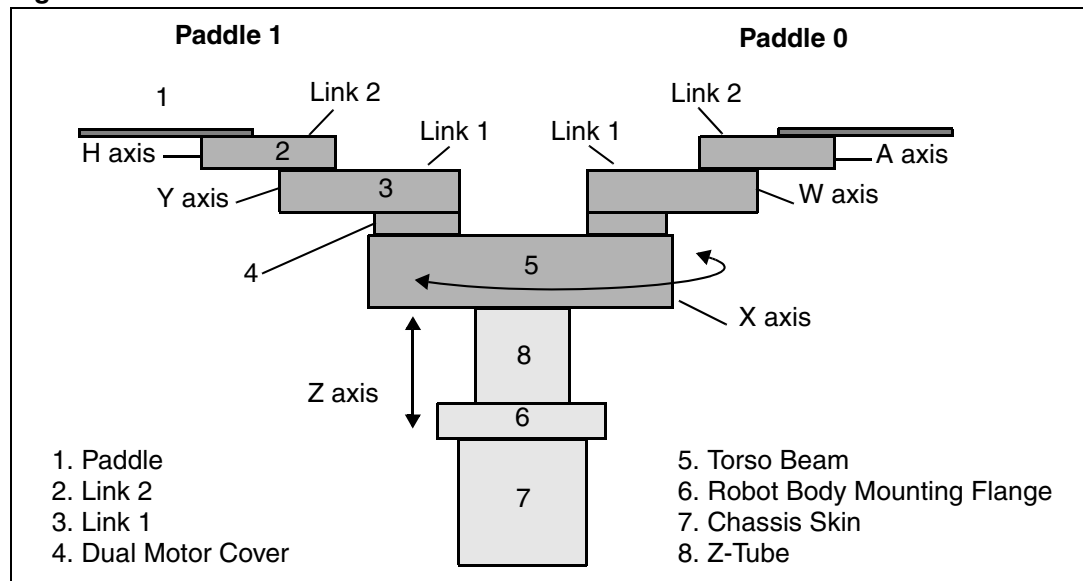
A significant issue in kinematic analysis surrounds so-called singular configurations. Physically, these configurations correspond to situations where the robot joints have been aligned in such a way that there is at least one direction of motion (the singular direction[s]) for the end effector that physically cannot be achieved by the mechanism. This occurs at workspace boundaries, and when the axes of two (or more) joints line up and are redundantly contributing to an end effector motion, at the cost of another end effector DOF being lost.

1.2 The industrial wafer handler

This section describes the main characteristics of an industrial wafer handler, used as a case study for a multiaxial planar manipulator family.

The structure of the manipulator consists of two arms with six joints (one prismatic joint and five revolute joints) arranged in order to have the motion axes parallel to each other.

The illustration below shows the relation between the axis control and the robot components.

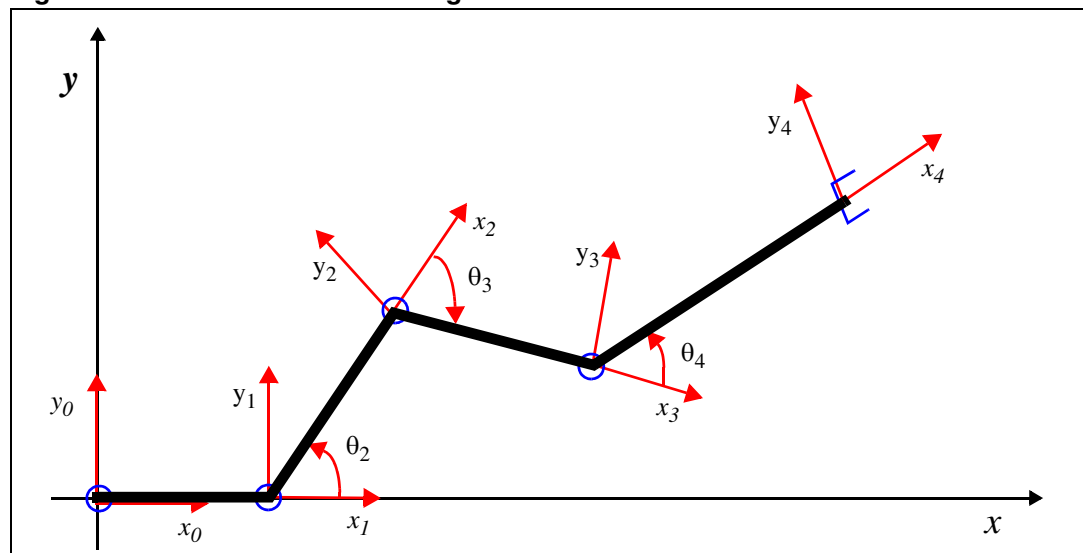
Figure 2. The structure of the wafer handler

The robot uses 6 electric motors placed as follows:

- Z Axis Motor (Vertical elevation)
- X Axis Motor (Rotation entire robot)
- Dual Motors (Link 2 and Link 1)

1.2.1 Forward kinematics

Accordingly to the Denavit-Hartenberg convention, an orthonormal cartesian coordinate system (x_i, y_i, z_i) has been established for each link, in order to characterize the forward kinematics through a D-H homogeneous transformation matrix.

Figure 3. The Denavit Hartenberg convention for one arm

Regarding the base coordinates (x_0, y_0, z_0), the z_0 axis lies along the axis of motion of the prismatic joint. The origin O_0 of the base coordinates has been placed at the same height of

the end effector, when the prismatic joint is at its lower level. The x_0 axis lies along the longitudinal direction of the link 1.

Since all the motion axes are parallel to each other, the origins of the other coordinate frames have been placed at the same height of the origin O_0 .

The Denavit-Hartenberg parameters are described in the following table:

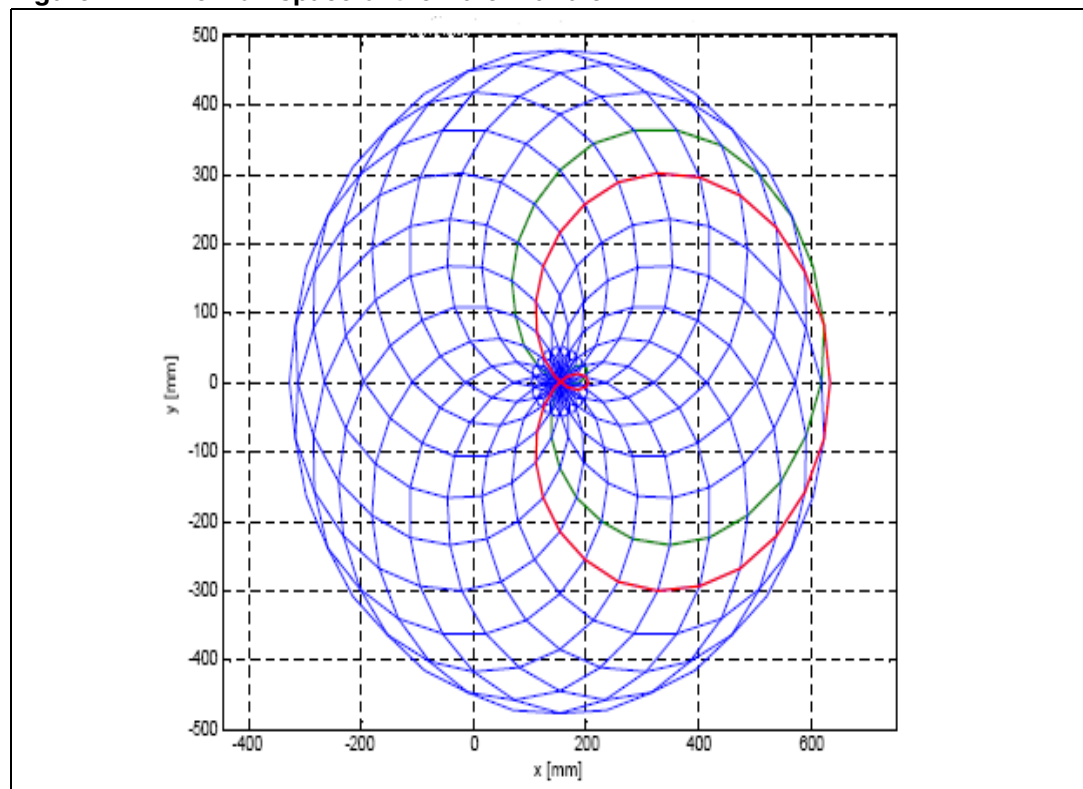
Table 2. The Denavit Hartenberg parameters for one arm

Link	a_i [mm]	α [rad]	d_i [mm]	θ [rad]
1	154	0	d1	0
2	133	0	0	2
3	133	0	0	3
4	215	0	0	$4 = -3/2$

In this application note we refer only to one arm, without considering the Z Axis (Vertical elevation - prismatic joint) and X Axis (Rotation entire robot - revolute joint).

Through a motion simulation it has been possible to obtain the workspace of the manipulator: for each value of the angle of the joint 1, the joint 2 performs a complete revolution. As shown in the [Figure 4](#), the workspace has a circular form. Nevertheless because the joint 3 is under-actuated, there are some regions of the workspace characterized of different orientations of the end-effector.

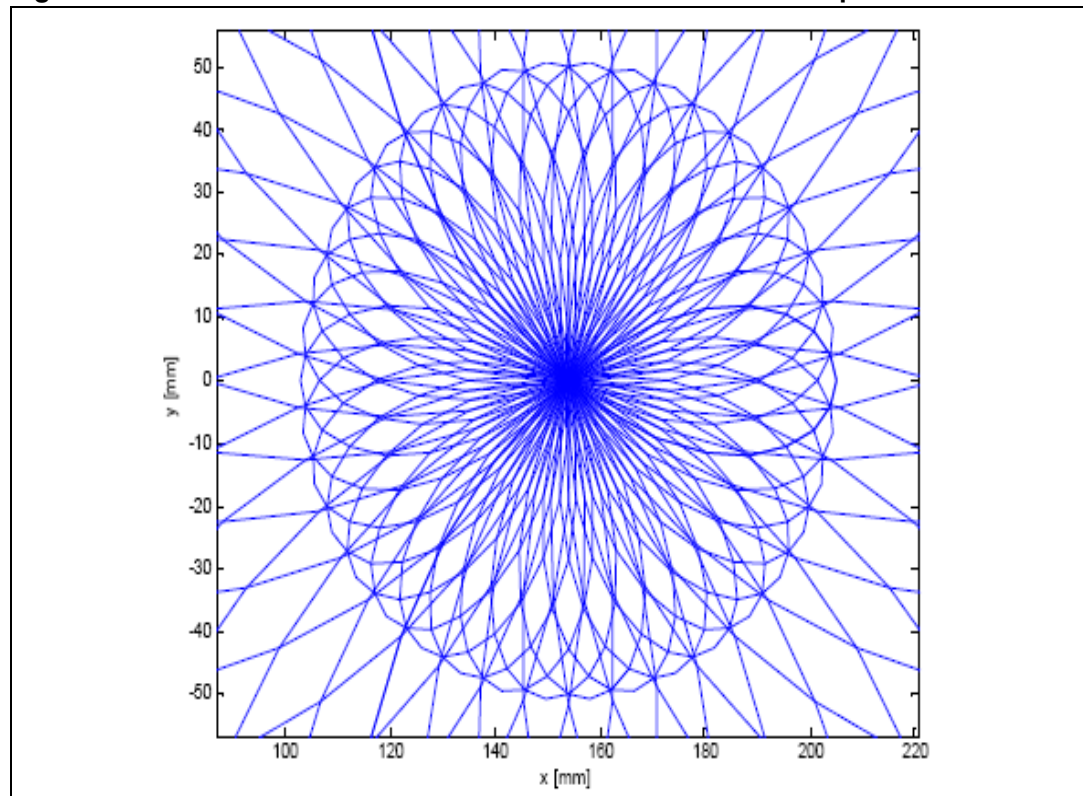
Figure 4. The workspace of the wafer handler



Out from the circumference, with origin (154 mm, 0 mm) and radius 51 mm, the manipulator is able to reach a point of the workspace with two possible configurations of the joint variables; the points inside this circumference are reachable with four possible configurations of the joint variables as shown in [Figure 5](#).

The center of the workspace is reachable with every orientation.

Figure 5. A detail of the central area of the wafer handler workspace

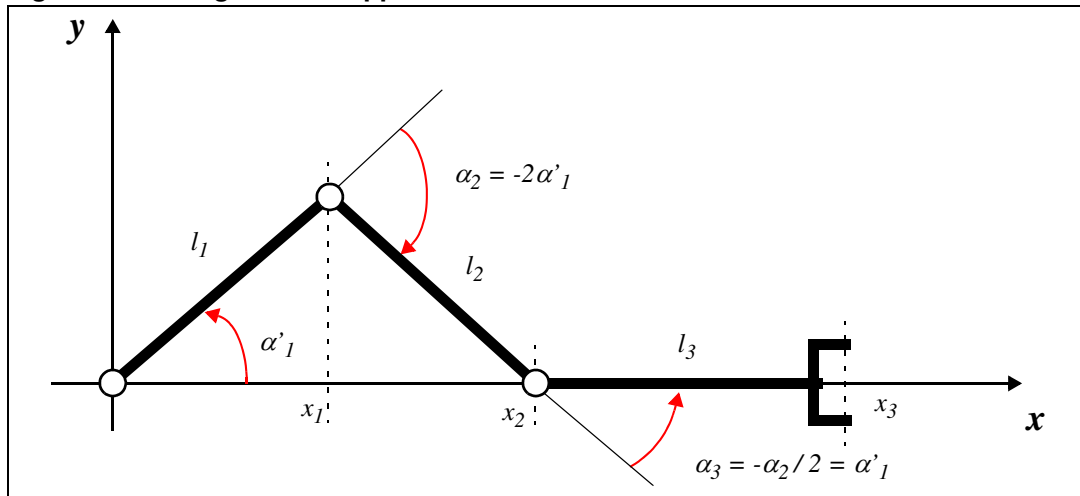


1.2.2 Inverse kinematics

A geometric approach based on the link coordinate systems and on the manipulator configuration has been used for the manipulator. Since we refer to the control of only one arm (link 1 - shoulder, link 2 - elbow), the origin of the base coordinates has been placed in such way that the z_0 axis lies along the rotation axis of the revolute joint of the shoulder.

First of all, we consider the manipulator in the configuration described in the following figure.

Figure 6. The geometric approach for the inverse kinematics



There is a mechanical connection between the link 3 and the link 2, which imposes $\alpha_3 = -\frac{\alpha_2}{2}$. This implies that the joint 3 can not be actuated independently from the joint 2.

Since $l_1 = l_2 = l$

$$\begin{cases} x_1 = l \cos \alpha'_1 \\ x_2 = x_1 + l \cos \alpha_2 = 2l \cos \alpha'_1 \\ x_3 = x_2 + l_3 = 2l \cos \alpha'_1 + l_3 \end{cases}$$

From the last equation and from geometric considerations follows:

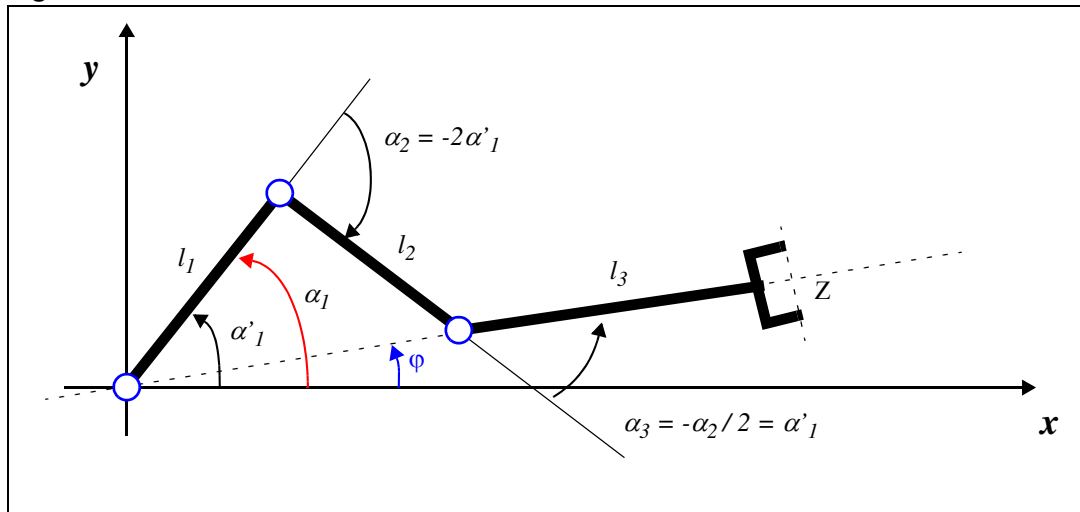
$$\alpha'_1 = \arccos \frac{x_3 - l_3}{2l}$$

$$\alpha_2 = -2\alpha'_1$$

$$\alpha_3 = -\frac{\alpha_2}{2} = \alpha'_1$$

Consider now a rotation of an angle φ :

Figure 7. The inverse kinematics after a rotation



Knowing the position of the end-effector in the workspace, it is possible to deduce its distance from the origin and the angle φ :

$$z = \sqrt{x^2 + y^2}$$

$$\varphi = \operatorname{atan} \frac{y}{x}$$

$$\alpha_1 = \varphi + \arccos \frac{z - l_3}{2l}$$

while α_2 and α_3 remains the same, because they are independent from the rotation of the manipulator.

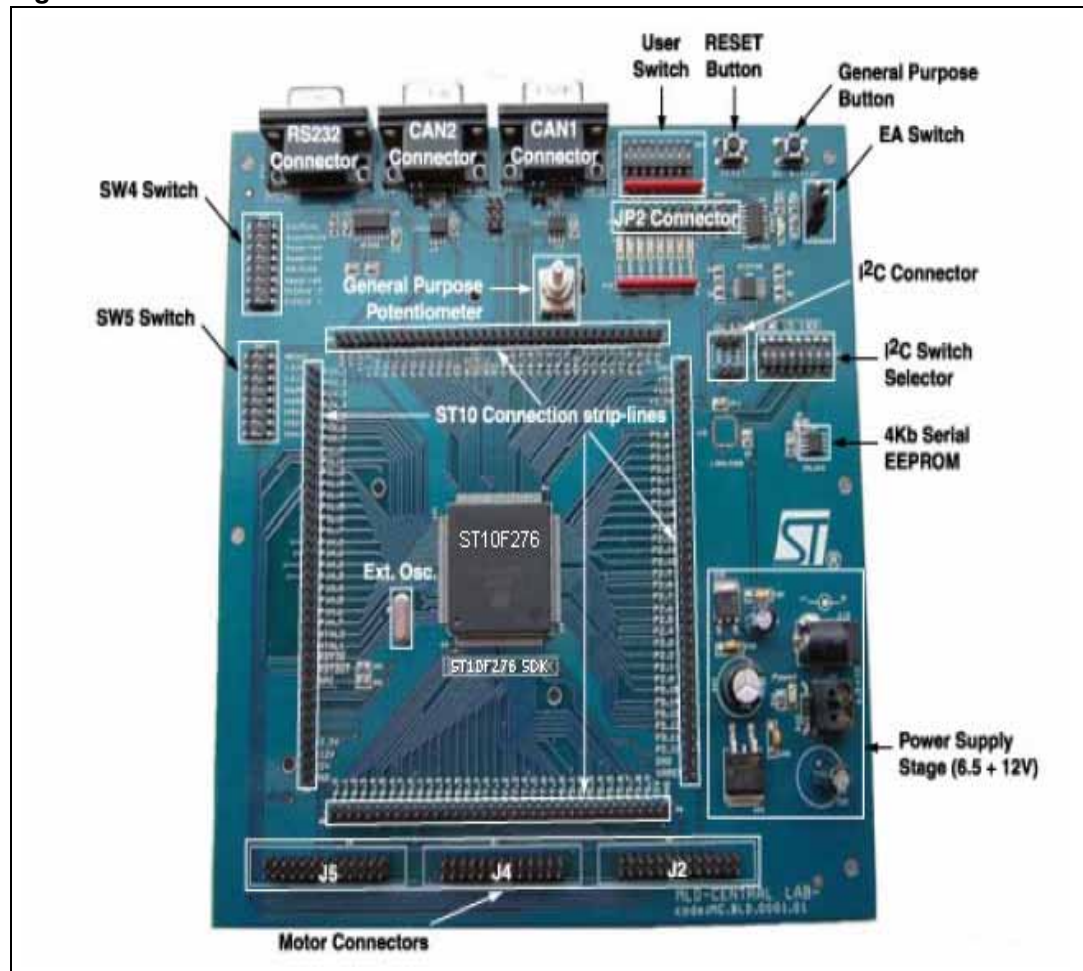
2 The SDK-ST10F276 control board

2.1 Brief description of the SDK-ST10F276

The SDK-ST10F276 is a two-layer, low-cost development board with an ST10F276 16-bit Embedded Flash Memory Microcontroller (see [Figure 8](#)). It is considered a general purpose application board used for developing advanced motor control solutions and processing external data (e.g., sensor outputs).

Three dedicated connectors for motion control allow the user to control different kinds of motors by plugging in external power motor boards.

Figure 8. The SDK-ST10F276 board



2.1.1 User Interfaces

The main interfaces used to communicate with the SDK-ST10F276 platform include:

- Two CAN 2.0B interfaces which operate on one or two CAN buses (30 or 2x15 message objects)
- One RS232 connector for serial signal communication with external devices
- One I2C connector for additional external device management
- Three dedicated connectors for motion control (external power motor boards)
- A series of general purpose LEDs and DIP switches
- One potentiometer and a push-button

For the user's convenience, the ST10 I/O pins on the Development board are split into four main connectors located on each side of the ST10 device

The five general purpose Light Emitting Diodes (LEDs) may be used for end-user applications.

They can be activated via the JP2 connector, or by using a series of dip switches on the board.

The potentiometer and push-button may be set in different and independent ways for general purpose uses.

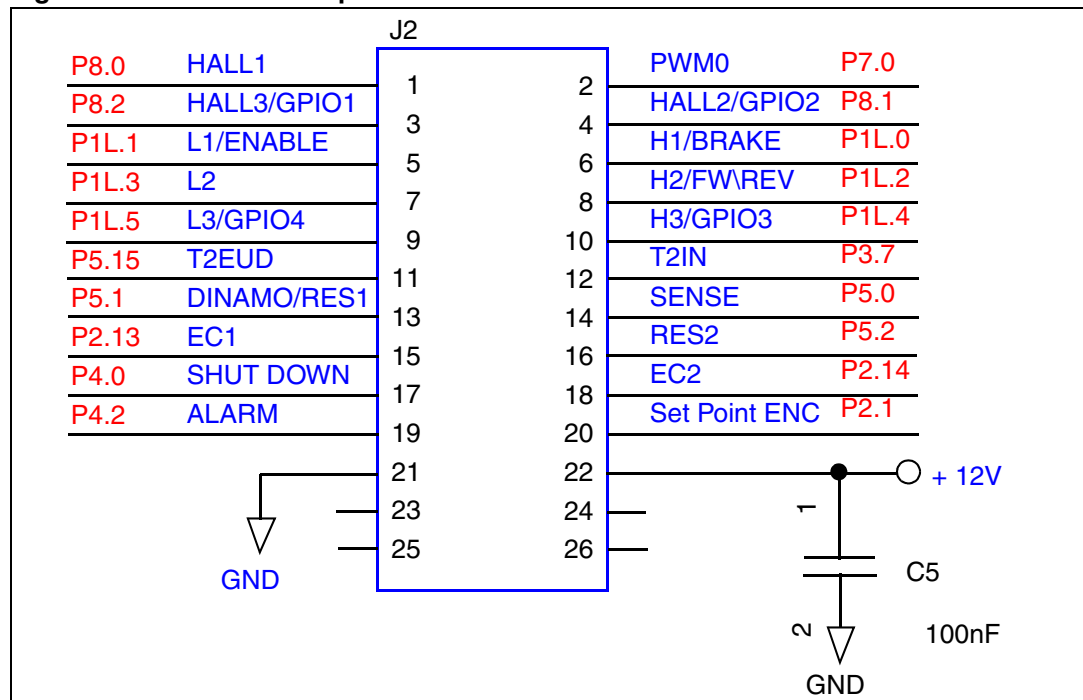
2.1.2 On board motor control connectors

The SDK-ST10F276 board provides an embedded solution (3 modular connectors) for power requirements up to 3000W, so they are well-suited for driving motors (e.g. BLDC, Brushless Direct Current) via an appropriate external motor board. The various power boards that can be connected to the SDK-ST10F276 are optimized in order to support a wide variety of advanced motor control applications.

Each of the three power connectors on the SDK-ST10F276 development board has 26 pins (see [Figure 9](#)). When a connector is used with a compatible motor board, the user can drive motors with the following signals:

- PWM
- Hall Effect sensor/GPIO
- Power/Brake
- Encoder
- Tachometer/Resolver
- Shutdown, and Alarm

Figure 9. One of the 26 pin connectors of the SDK-ST10F276 board

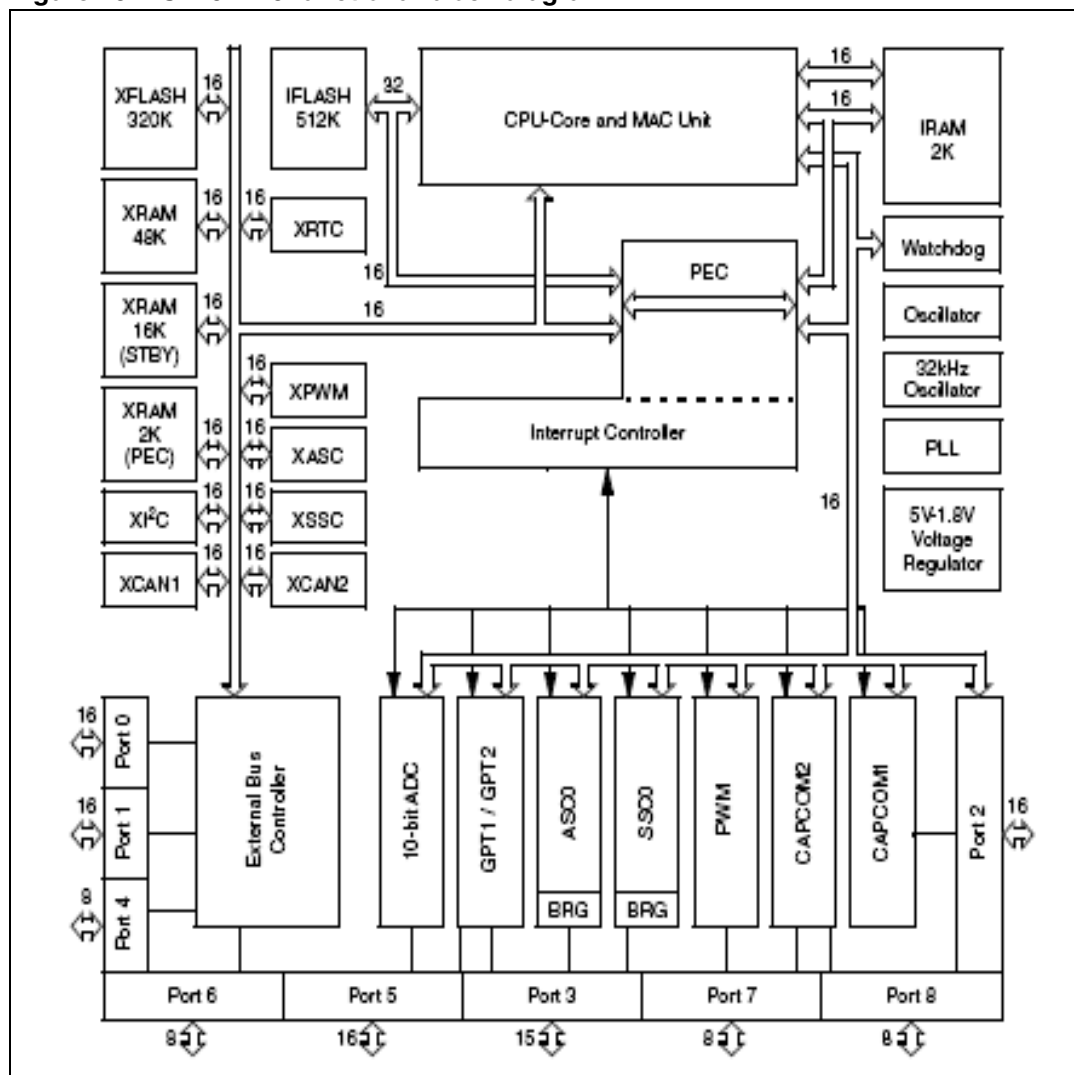


2.2 ST10F276 16-bit microcontroller - architectural overview

The ST10F276 is a high performance 64MHz 16-bit microcontroller with DSP functions.

ST10F276 architecture combines the advantages of both RISC and CISC processors with an advanced peripheral subsystem. The following block diagram gives an overview of the different on-chip components and of the advanced, high bandwidth internal bus structure of the ST10F276. (see [Figure 10](#)).

Figure 10. ST10F276 functional block diagram



2.2.1 Basic CPU concepts

The main core of the CPU includes a 4-stage instruction pipeline, a 16-bit arithmetic and logic unit (ALU) and dedicated SFRs.

Several areas of the processor core have been optimized for performance and flexibility. Functional blocks in the CPU core are controlled by signals from the instruction decode logic. The core improvements are summarized below:

- High instruction bandwidth / fast execution
- High function 8-bit and 16-bit arithmetic and logic unit
- Extended bit processing and peripheral control
- High performance branch, call, and loop processing
- Consistent and optimized instruction formats
- Programmable multiple priority interrupt structure

2.2.2 Memory organization

The memory space of the ST10F276 is organized as a unified memory. Code memory, data memory, registers and I/O ports are organized within the same linear address space. The main characteristics are summarized below:

- 512K Bytes of on-chip single voltage Flash memory with Erase/Program controller
- 320K Bytes of on-chip extension single voltage Flash memory with Erase/Program controller (XFLASH)
- Up to 100K Erase/Program cycles
- Up to 16 MB of linear address space for code and data (5 MB with CAN or I²C)
- 2K Bytes of on-chip internal RAM (IRAM)
- 66K Bytes of on-chip extension RAM (XRAM)

2.2.3 On-chip peripheral blocks

The ST10F276 generic peripherals are:

- Two General Purpose Timer Blocks (GPT1 and GPT2)
- Serial channel: Two Synchronous/Asynchronous, two High-Speed Synchronous, I²C standard interface
- A Watchdog Timer
- Two 16-channel Capture / Compare units (CAPCOM1 and CAPCOM2)
- A 4-channel Pulse Width Modulation unit and 4-channel XPWM
- A 10-bit Analog / Digital Converter
- Up to 111 General Purpose I/O lines

Each peripheral also contains a set of Special Function Registers (SFRs), which control the functionality of the peripheral and temporarily store intermediate data results. Each peripheral has an associated set of status flags. Individually selected clock signals are generated for each peripheral from binary multiples of the CPU clock.

2.2.4 Managing Interrupts (hardware)

- 8-channel Peripheral Event Controller for single cycle, interrupt-driven data transfer
- 16-levels of priority for the Interrupt System with 56 sources, and a sampling rate down to 15.6ns at 64MHz

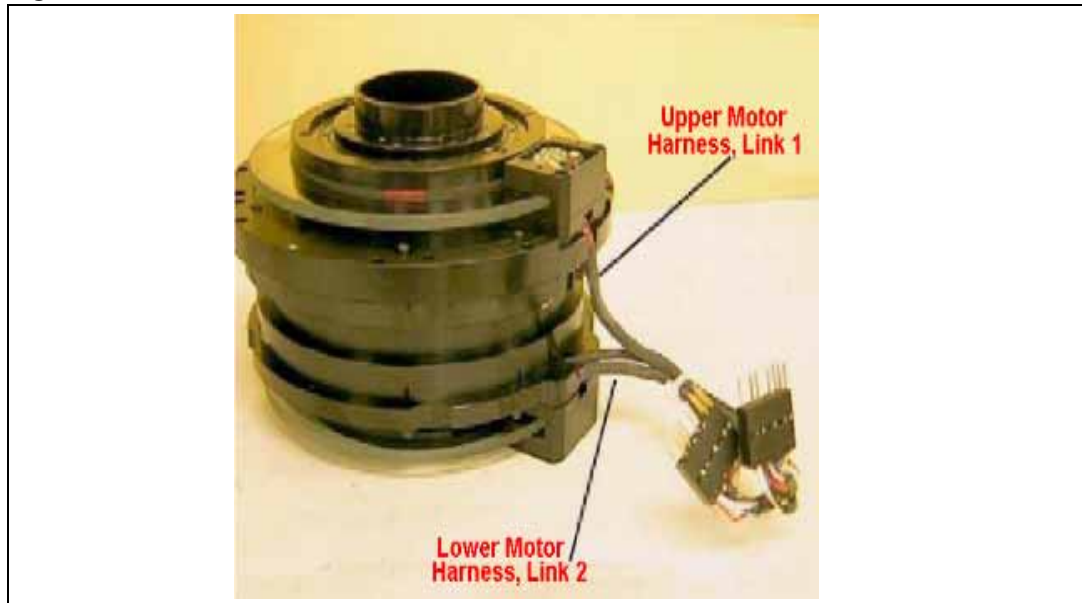
Note: Refer to the ST10 information listed in the [Introduction on page 1](#) for device details.

3 Hardware and mechanical equipments

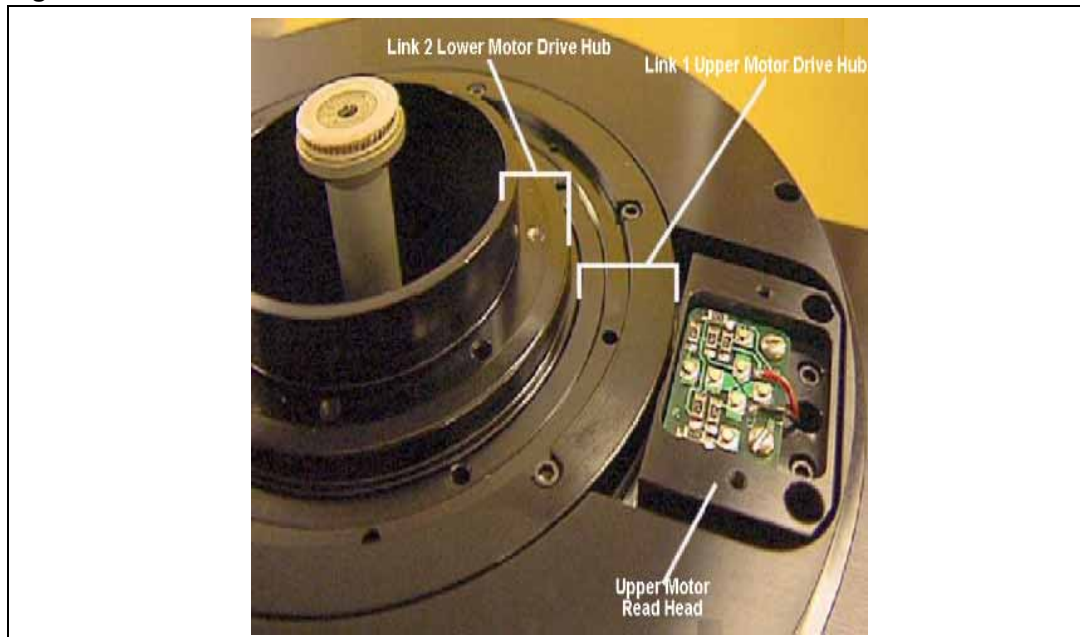
3.1 The Dual DC motor and the power stage

Each arm of the wafer handler is controlled by a dual DC motor (see [Figure 17 on page 24](#)); they are mounted in the same chassis with two armatures which rotate on the same axis and having concentric drive hubs, as shown in the [Figure 12](#).

Figure 11. The Dual DC motor



The upper armature is connected to the outer drive hub and controls the shoulder (link 1), while the lower armature is connected to the inner drive hub and controls the elbow (link 2) through a drive belt with a transmission ratio of 1/2.

Figure 12. The chassis of the dual DC motor

Each DC motor is driven by a monolithic solution (L6205PD) that is a dual full bridge driver. In order to increase the current capability, the two full bridges inside the driver have been connected in parallel mode.

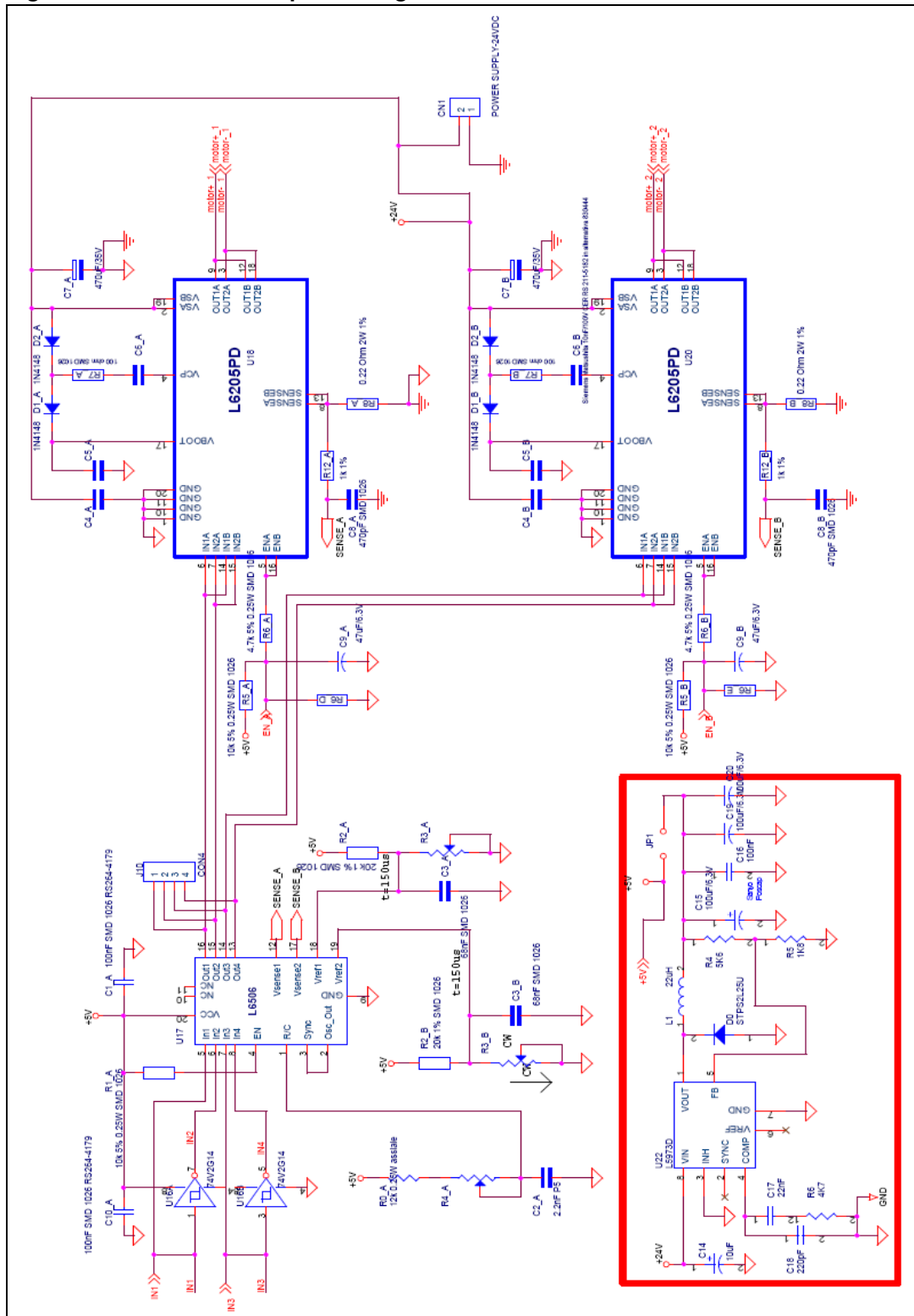
A current controller IC (L6506) has been used for both the monolithic drivers in order to sense the current motor. When the sensed motor current exceeds its current rate, the current controller disables the power drivers.

Each motor is driven through a PWM signal generated by the microcontroller; the velocity and the direction of rotation of each DC motor depends on the duty cycle of such signal. For duty cycles over 50%, the motor moves in one direction, while for duty cycles under 50%, the motor moves in the opposite direction. For a duty cycle of 50% the motor is stopped.

The driver board has been developed in order to control three links, even if the control algorithm refers to the control of only one arm (two links).

In the driver board there is also placed a DC/DC converter that generates all the needed voltages to supply all the circuits and the SDK-ST10F276 board. A detail of the schematic of the power stage is shown in the following figure.

Figure 13. A detail of the power stage



3.2 Cables and connectors

On the chassis skin of the wafer handler are present six 15 pin D-SUB male connectors. Four of them (named A, W, Y, H) are used for the links of the two arms, while the other two, named X and Z, are used respectively for the torso beam and the Z elevation. The four harness connectors of the links are located under the torso beam.

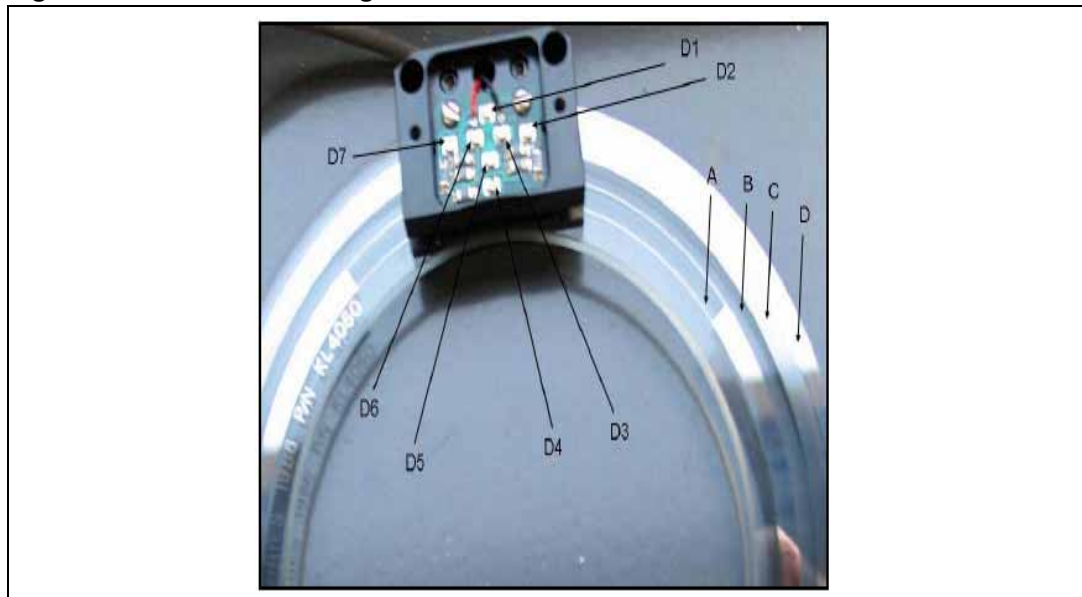
The wafer handler is furnished of cables, each of one presents on one side a 15 pin D-SUB female connector, and on the other side a 25 pin D-SUB female connector. The former is connected to the wafer handler, while the latter is connected to the driver board. The table below shows the pins assignment and the related description.

Table 3. The sectors of the absolute encoder

Pin DB15 Connector	Pin DB25 Connector	Wire	Description
1	1	Green	Incremental encoder (sin)
2	2	Orange	Incremental encoder (cos)
3	3	Violet	Absolute encoder (bit b_1)
4	16	Not used	
5	9	White	Absolute encoder (bit b_2)
6	10	Grey	Absolute encoder (bit b_0)
7	7	Not used	
8	8	Red	2.5V supply for encoder read head
9	14	Yellow	VREF-
10	15	Blue	VREF+
11	19	Not used	
12	21	Black	Ground for encoder read head
13	23	Not used	
14	24	Black (motor)	Motor +
15	25	White (motor)	Motor -

3.3 The encoders and the conditioning circuit

Each joint of the arm of the manipulator has an encoder ring as shown in [Figure 14](#). In this ring it is possible to identify four sector named A, B, C and D, and the encoder read head.

Figure 14. The encoder ring of one DC motor

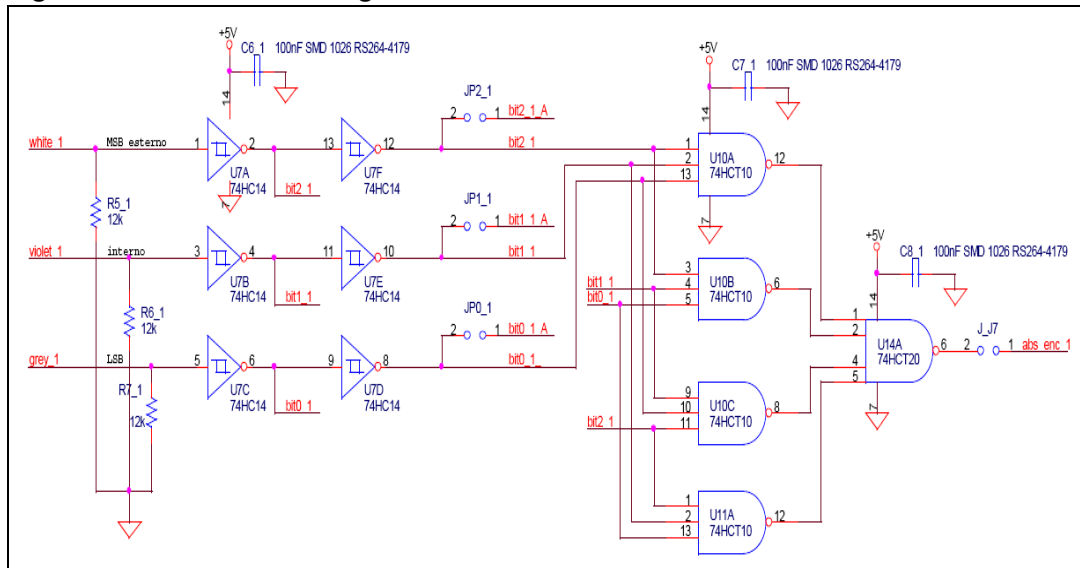
The sectors D, A and B represent the absolute encoder, while the sector C represents the incremental encoder.

The opaque parts of each sector of the absolute encoder are placed according to a Gray code, in order to offer absolute position of the link with a resolution of 45 degrees. In this way it is possible to identify 8 angular positions accordingly to the scheme below, where the sector D (white wire) represents the most significant bit of the encoder, while the sector A (grey wire) represents the least significant bit.

Table 4. The sectors of the absolute encoder

Angular position	Sector D (white wire) b_2	Sector B (Violet wire) b_1	Sector A (Grey wire) b_0	Absolute encoder values $b_2b_1b_0$
1	0	0	0	0
2	0	0	1	1
3	0	1	1	3
4	0	1	0	2
5	1	1	0	6
6	1	1	1	7
7	1	0	1	5
8	1	0	0	4

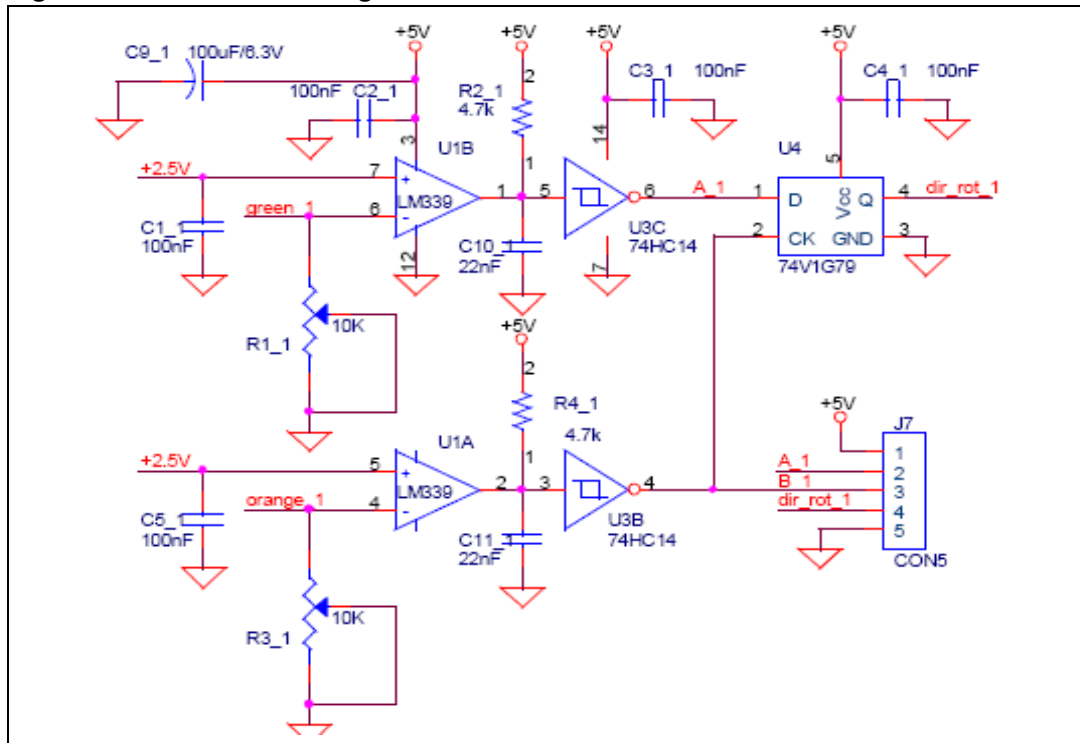
In order to capture every transition, in each bit of the absolute encoder has been developed a logic circuit with NAND ports. A detail of the conditioning circuit for one of the absolute encoders of the wafer handler is shown in [Figure 15](#). In this way the output of this circuit can be associated to an external interrupt of the microcontroller, while the output bits of the absolute encoder can be associated to three GPIOs (General Purpose Input Output) in order to read its value.

Figure 15. The conditioning circuit for one absolute encoder

The sector C represents the incremental 12-bit encoder, whose outputs are two sinusoidal signals (available in orange and green wires) shifted to each other by 90 degrees, with an offset of 2.5 V and a peak-to-peak voltage of 1.1 V.

Since these signals can not be directly managed by the microcontroller, it has been necessary to develop a conditioning circuit whose outputs are digital signals.

This conditioning circuit for each sinusoidal signal consists of a comparator whose output is at a logical high level (5 V) every time the input signal level is over 2.5 V, and at logical low level (0 V) every time the level of the input signal is below 2.5 V. A detail of the conditioning circuit for one of the incremental encoders of the wafer handler is shown in the following figure.

Figure 16. The conditioning circuit for one incremental encoder

Using a trimmer it is possible to have a square wave with a duty cycle of 50% as output.

In this way from the two input sinusoidal signals it is possible to obtain two square signals shifted each other of 90 degrees, through which it is possible to count 4096 pulses for each rotation of the joint.

The conditioning circuit presents also a D flip-flop through which it is possible to obtain the direction of rotation of the joint.

3.4 Schematics of the driver board and the interface board.

The driver board presents three 25 pin D-SUB male connectors to which are plugged the cables of the wafer handler. In the same board there are 26 pin male connectors, used to interface the SDK-ST10F276 board through three flat cables.

Figure 17 and *Figure 18* show the schematic of the driver board.

Figure 17. Driver board schematic (page 1)

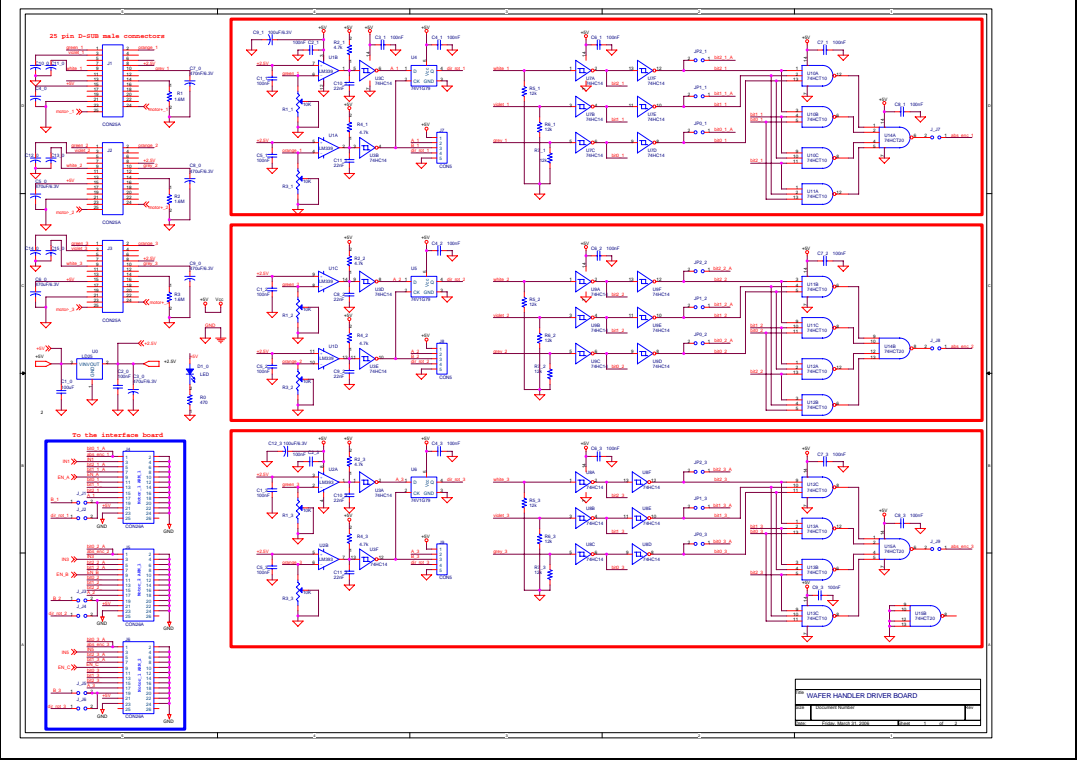
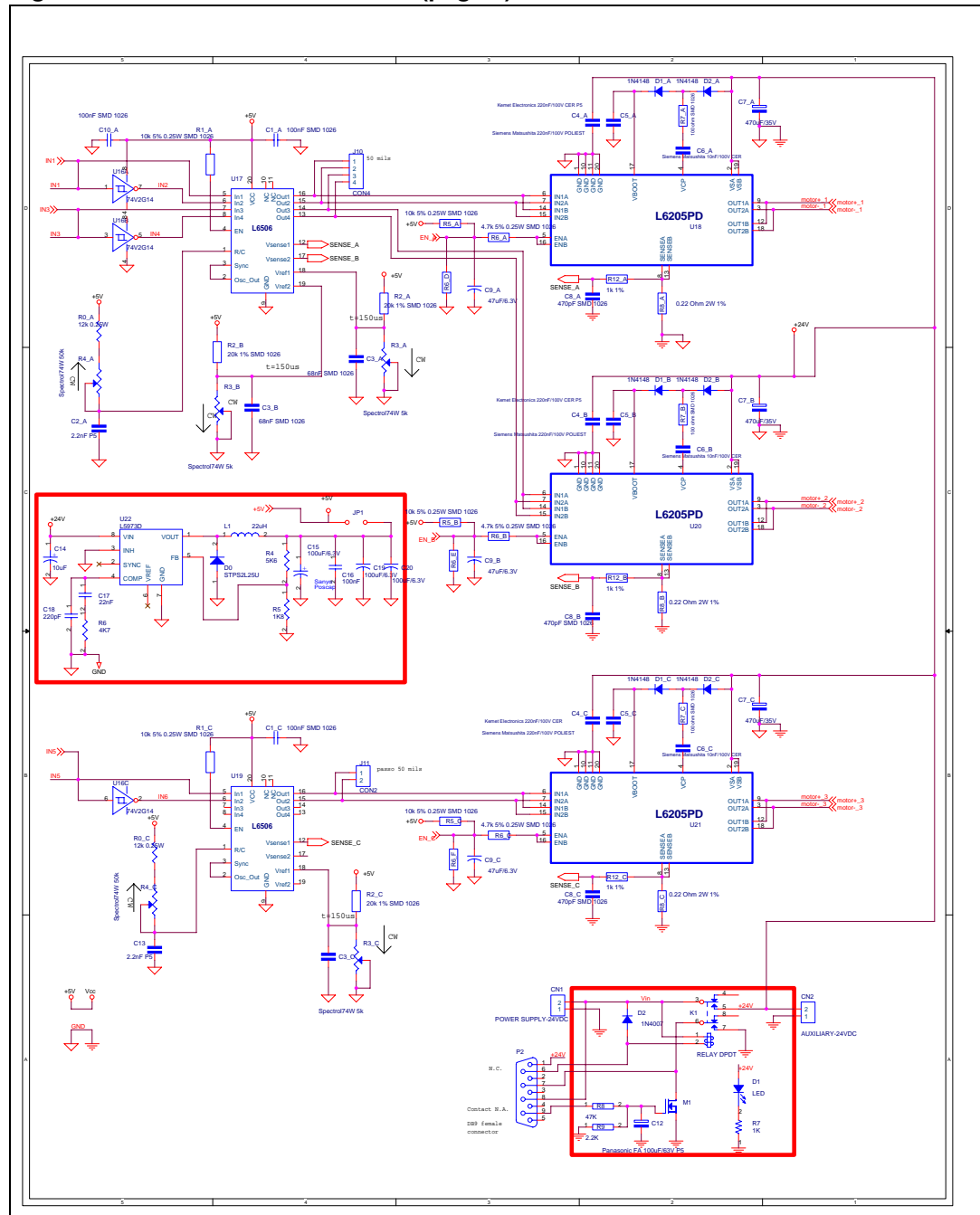
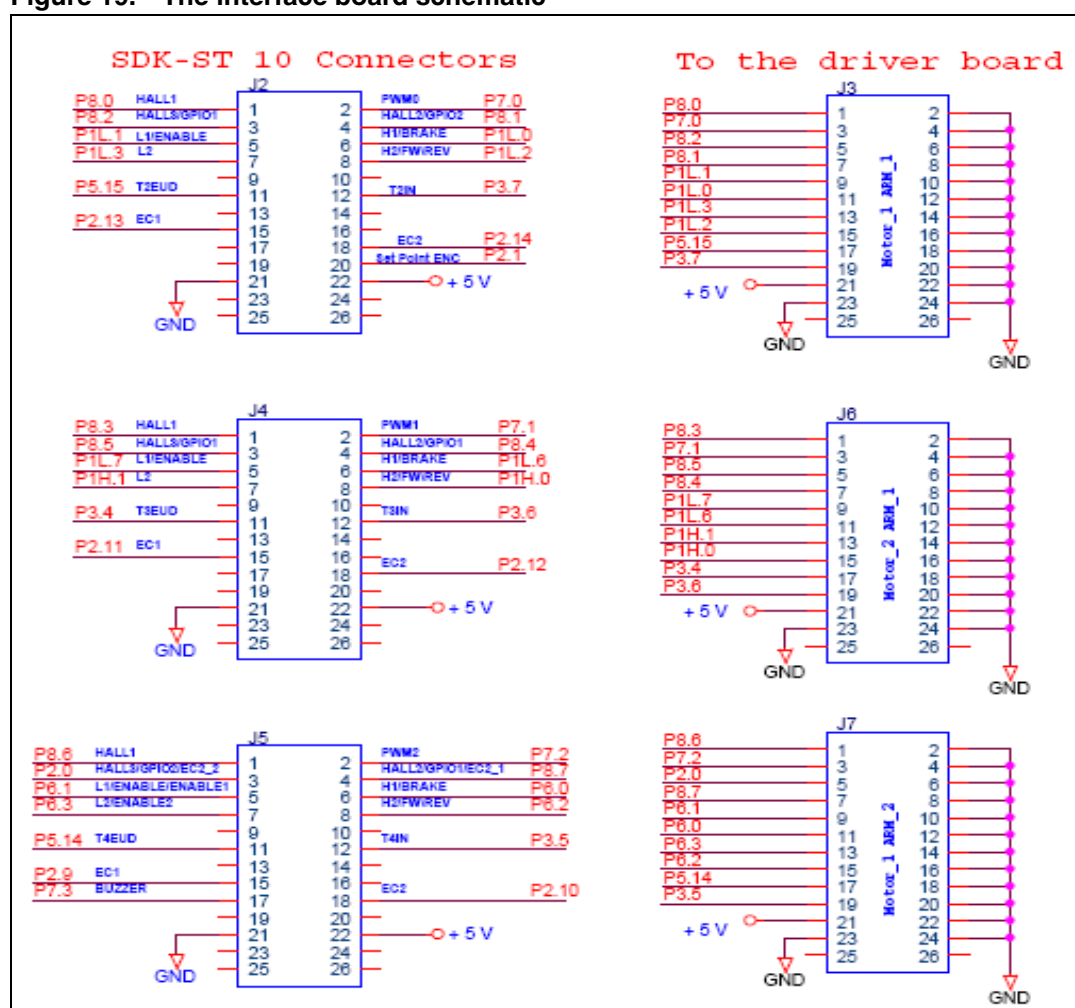


Figure 18. Driver board schematic (page 2)



To avoid electromagnetic interference, all the signals have been shielded. For this reason an interface board with three 26 pin female shielded connectors has been developed, in order to be plugged in the male connectors of the SDK-ST10F276 board. The schematic of the interface board is shown below.

Figure 19. The interface board schematic



4 Control algorithm

4.1 Motion and path planning

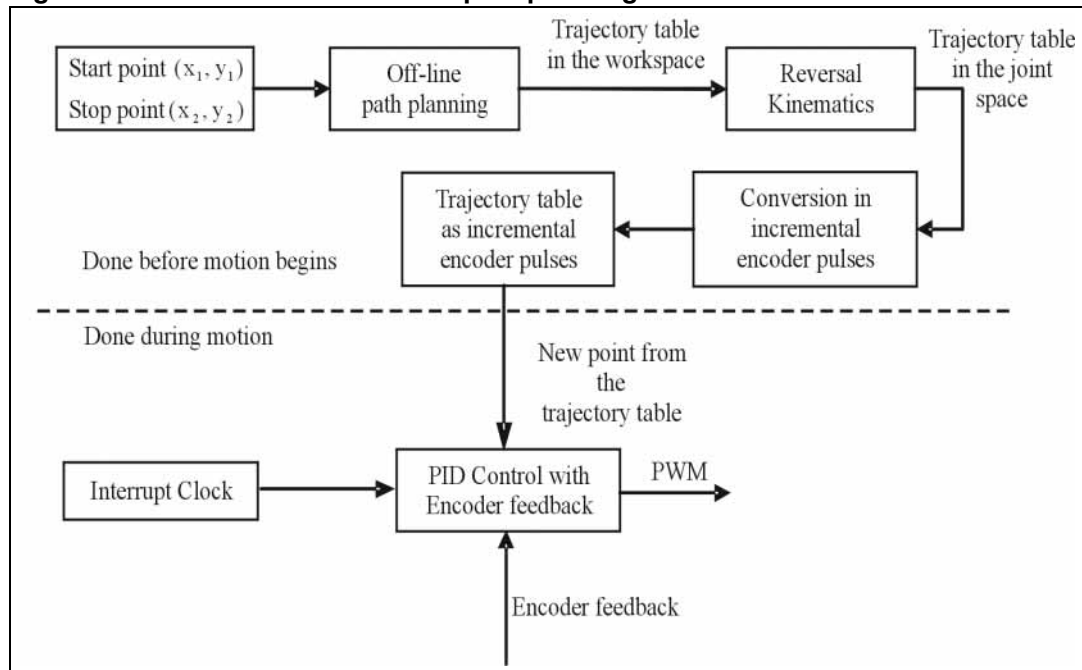
Path planning problems generally involve computing a continuous sequence (a *path*) of configurations (generalized coordinates) between an initial configuration (start) and a final configuration (goal) respecting certain constraints.

The term *motion planning* is usually distinguished from *path planning* in that the computed path is parameterized by time (i.e. a trajectory). The consideration of *time* is often important for problems requiring time-parameterized solution trajectories.

The aim of the motion planning is to generate the inputs of the motion control system in order to ensure the correct execution of the planned trajectory by the manipulator.

The path planning can be realized in the joint space or in the workspace. The latter has been used in this work according to the scheme of the following figure.

Figure 20. General scheme for the path planning

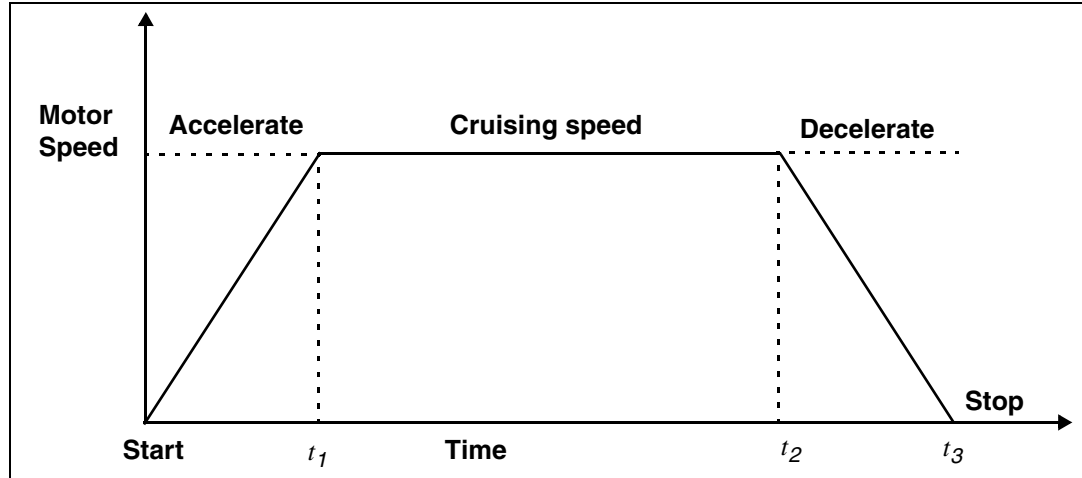


As can be seen, all the reference points necessary to the control of the desired trajectory are calculated off-line. The trajectory planning algorithm generates a temporal sequence of variables that specify the position and orientation of the end-effector in the workspace. Because the control of the manipulator is realized as control of the joints, from this sequence of variables it is necessary to perform the kinematics inversion to build a sequence of variables in the joint space. These joint variables must be opportunely converted in incremental encoder pulses, because the position closed loop control is performed on the feedback of the incremental encoders of the two joints.

The path planning method used in this work is the *straight-line motion*. In this method the end-effector of the robot travels in a straight line between the start and stop points, using a trapezoidal velocity profile, in which the motion controller must command the motor driver to

gradually ramp-up the motor velocity, until it reaches the desired speed and then, gradually ramp-down it until it stops after the task is complete, as shown in [Figure 21](#):

Figure 21. Example of trapezoidal velocity profile



The trapezoidal velocity profile is analytically expressed as follows:

$$v(t) = \begin{cases} at & 0 \leq t < t_1 \\ v_c & t_1 \leq t < (t_3 - t_1) \\ v_c - a(t - t_3 + t_1) & (t_3 - t_1) \leq t < t_3 \end{cases}$$

where t_1 is the time slot for the ramp-up, t_3 is the total time for the execution of the entire trajectory, and $t_2 = (t_3 - t_1)$ is the time slot during which the end effector moves at desired speed v_c (also called the cruising speed).

Integrating it respect to the time, it is possible to obtain the function that describes the space covered in the workspace.

$$s(t) = \begin{cases} \frac{1}{2}at^2 & 0 \leq t < t_1 \\ v_c(t - t_1) + \frac{1}{2}at^2 & t_1 \leq t < (t_3 - t_1) \\ v_c(t_3 - 2t_1) + \frac{1}{2}at_1^2 + v_c(t - t_3 + t_1) - \frac{1}{2}a(t - t_3 + t_1)^2 & (t_3 - t_1) \leq t < t_3 \end{cases}$$

The time t_1 can be obtained from the knowledge of the cruising speed and the acceleration a (that represent the known parameters)

$$t_1 = \frac{v_c}{a}$$

and the total time t_3 can be obtained from the distance s_{max} between the start point and the stop point as follows:

$$s_{max} = s(t_3) = v_c t_3 - v_c t_1 \Rightarrow t_3 = \frac{s_{max}}{v_c} + t_1$$

Now follows the algorithm used for the motion planning in order to build a trajectory table whose points (as joint variables after the reversal kinematics and expressed as incremental encoder pulses) are given as inputs to the PID control interrupt service routine every T_R , (fixed in 20 ms).

Let (x_1, y_1) the coordinates of the start point and (x_2, y_2) the coordinates of the stop point, v_c the cruising speed and a the acceleration. From the previous equations it is possible to obtain the time t_1 and the time t_3 , where:

$$s_{\max} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Consider $y = mx + c$ as the generic straight-line that intersect the two points (x_1, y_1) and (x_2, y_2) , where

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

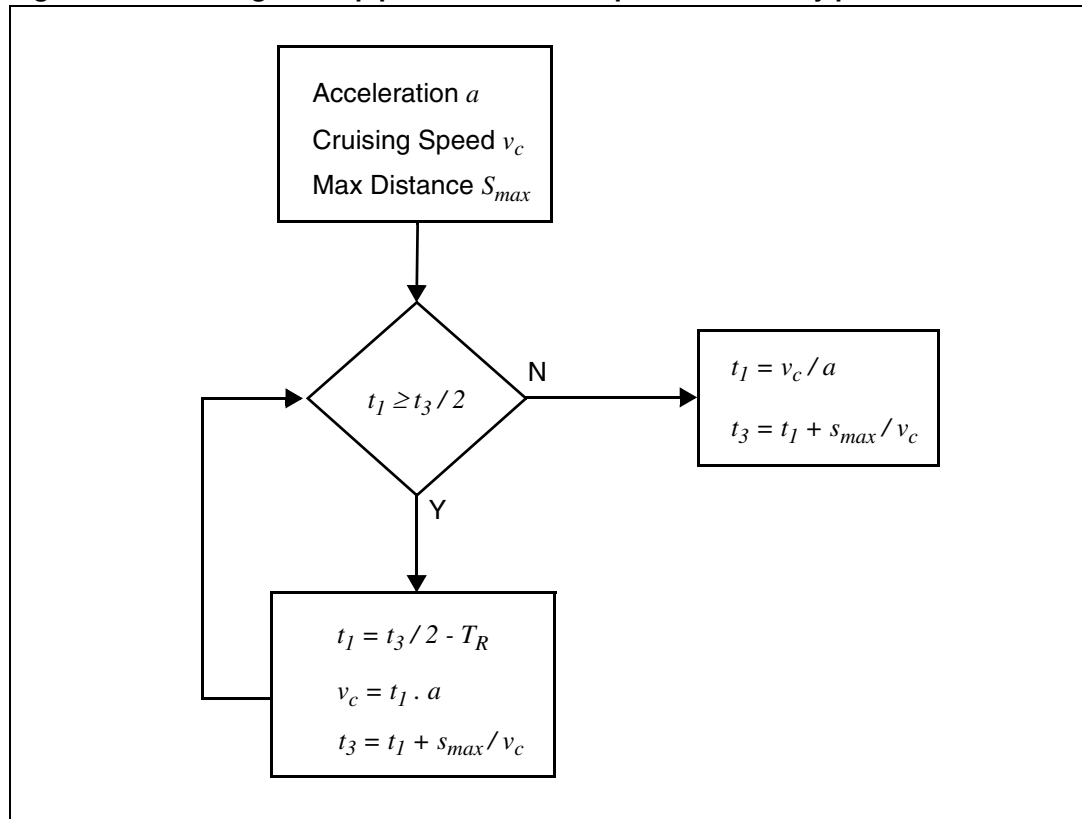
is the angular coefficient and

$$c = \frac{1}{2} \left(y_1 + y_2 - \frac{(y_2 - y_1)(x_2 + x_1)}{(x_2 - x_1)} \right)$$

is the intersection with the Y axis.

It is possible that the cruising speed it is not compatible with the imposed acceleration and the distance to cover: this implies that the velocity profile is not trapezoidal. For this reason a convergent loop procedure has been developed (shown in the following scheme) that redefines the values of t_1 , v_c , and t_3 .

Figure 22. Convergent loop procedure for a trapezoidal velocity profile



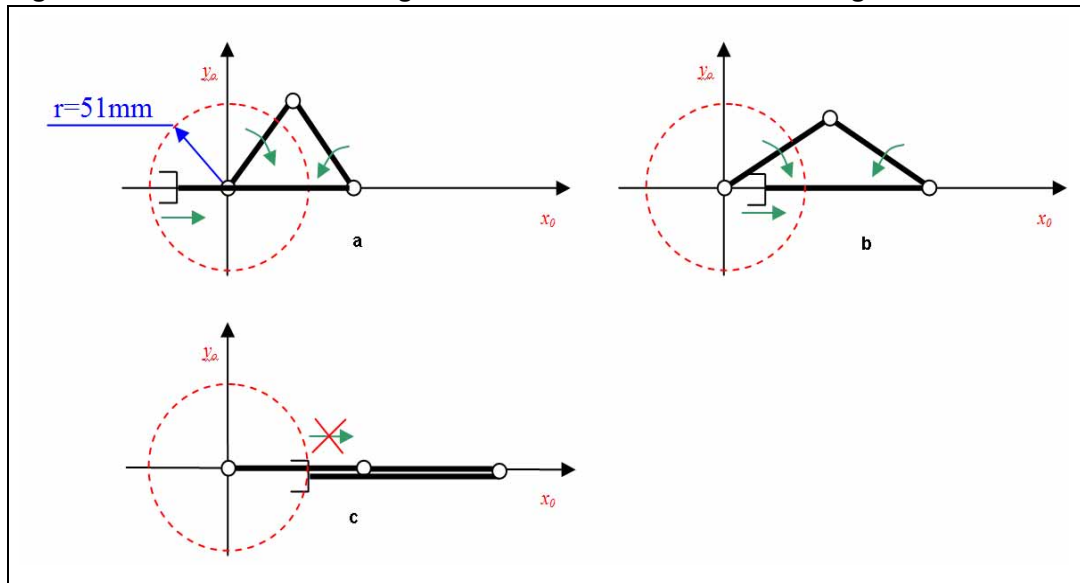
At the generic sampling time T_i , the covered distance, accordingly to the previous equation, is equal to $s_i = s(T_i)$; so the point (x, y) of the trajectory to reach in a straight-line motion at time T_i in the workspace, can be obtained resolving the following system:

$$\begin{cases} s_i^2 = (x - x_1)^2 + (y - y_1)^2 \\ y = mx + c \end{cases}$$

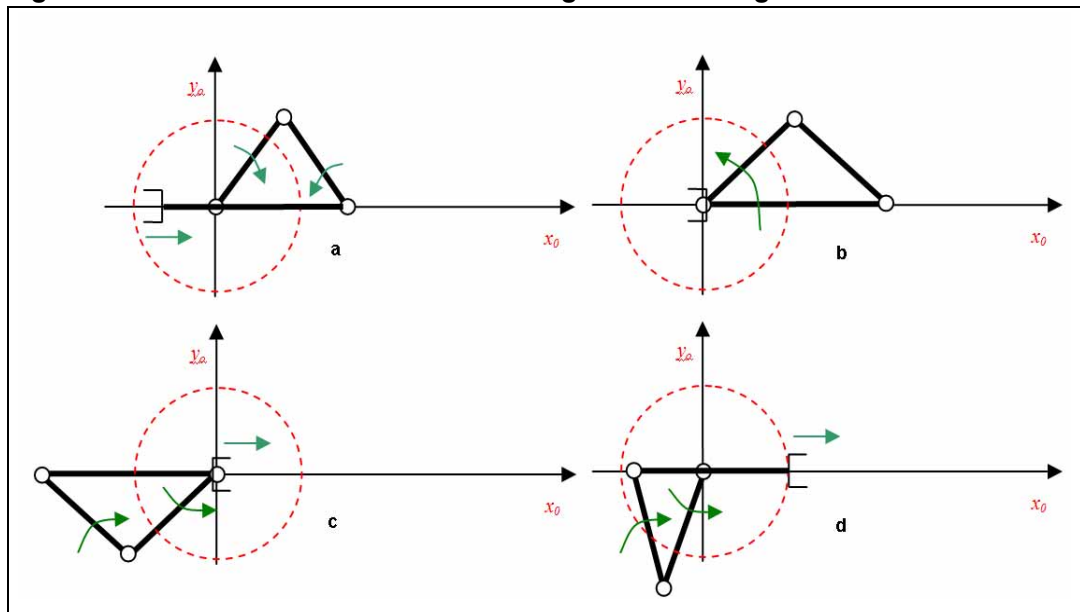
Knowing (x, y) it is possible to obtain the corresponding joint variables α_1 and α_2 with the inverse kinematics equations. These joint variables must be opportunely converted in incremental encoder pulses, because the position closed loop control is performed on the feedback of the incremental encoders of the two joints.

During the trajectory planning it is very important that the manipulator avoids crossing the kinematics singularities of the accessible workspace since they could make the continuation of the trajectory impossible.

For this wafer handler, these singularities are distributed along the circle with radius 51 mm and center in the origin of the base coordinates. The motion singularity appears only when the manipulator, during the motion, crosses the point of the circle assuming a configuration with the arms aligned, as shown in the following figure.

Figure 23. The kinematics singularities of the wafer handler during motion

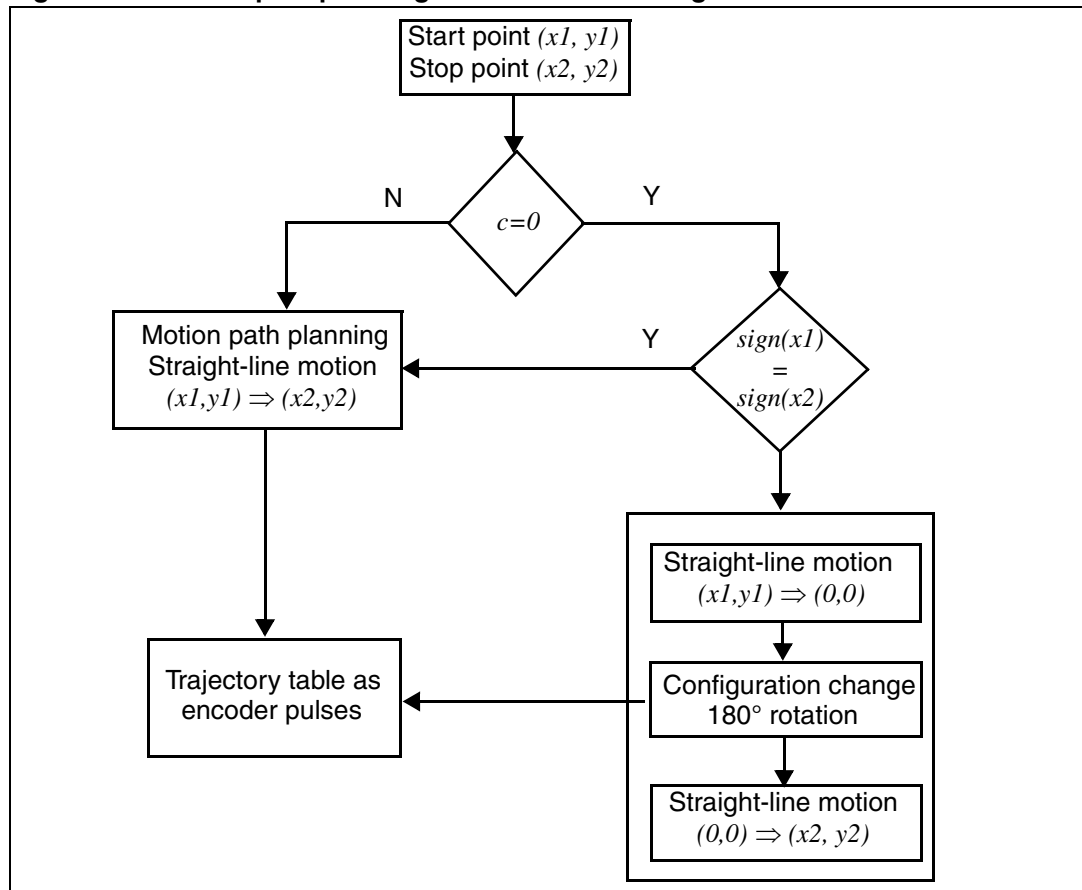
This problem happens only when the manipulator moves along a straight-line passing through the origin ($c = 0$) and when the abscissas x_1 and x_2 are of opposite sign. The problem has been solved making a change of orientation in the origin in order to cross the kinematics singularity with an alternative configuration as shown in [Figure 24](#).

Figure 24. How to solve the kinematics singularities during the motion

As shown in the figure above, the trajectory in this case is made of three tracts.

- A straight-line tract with a trapezoidal velocity profile that brings the end-effector from the start point (x_1, y_1) to the origin $(0,0)$.
- A change of the configuration in which the entire arm is rotated of 180° . We can see how the position of the end-effector remains the same while its orientation is changed.
- A straight-line tract with a trapezoidal velocity profile that brings the end-effector from the origin $(0,0)$ to the stop point (x_2, y_2) .

Figure 25. Motion path planning scheme to avoid singularities



4.2 PID position control algorithm

The points of the trajectory table have to be given as input one by one to the control algorithm every T_R , that has been fixed in 20 ms, while the PID control interrupt service routine (ISR) is performed every 1ms.

The trajectory table is composed of two arrays; one for the shoulder (*shoulder_array[]*) and one for the elbow (*elbow_array[]*), obviously of the same dimension.

The control ISR has been associated to the compare unit CC2, while the scanning of the trajectory table has been associated to the compare unit CC3.

The incremental encoders of the shoulder and the elbow have been associated to the GPT1 timers T2 and T4, respectively, used in incremental interface mode.

To ensure that the point has been reached, a check on the error has been performed in a such way that if the total error (as sum of the error of the shoulder and of the elbow) is higher than a fixed value (MAX_ERR_POS=150) no new point is passed. The figures below show the general schemes for the control algorithm ISR and for the scanning of the trajectory table.

The variables *sh_pulses_desired*, *elb_pulses_desired*, *total_error*, *index*, *flag_error* are global variables.

Figure 26. PID position control ISR

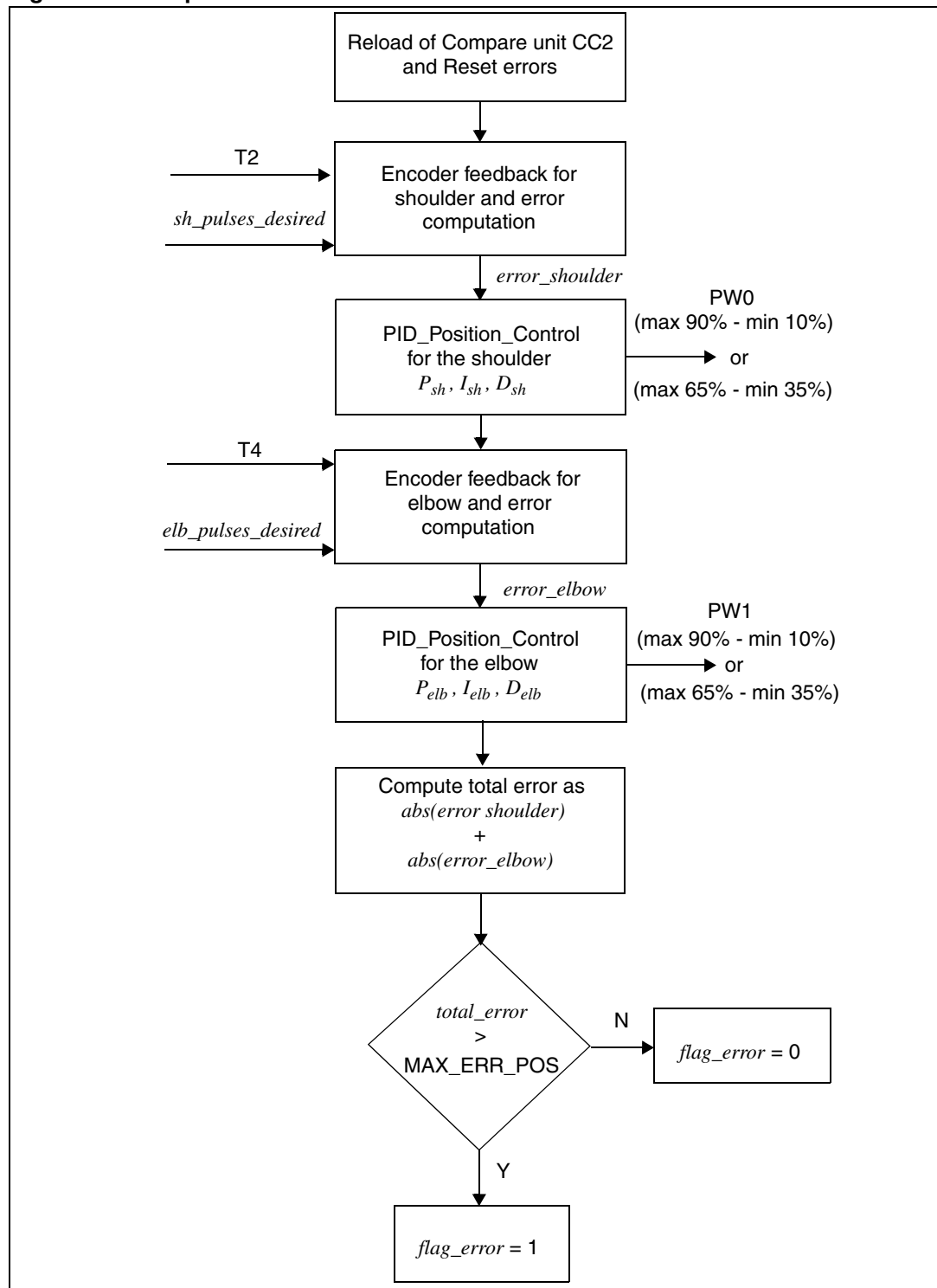
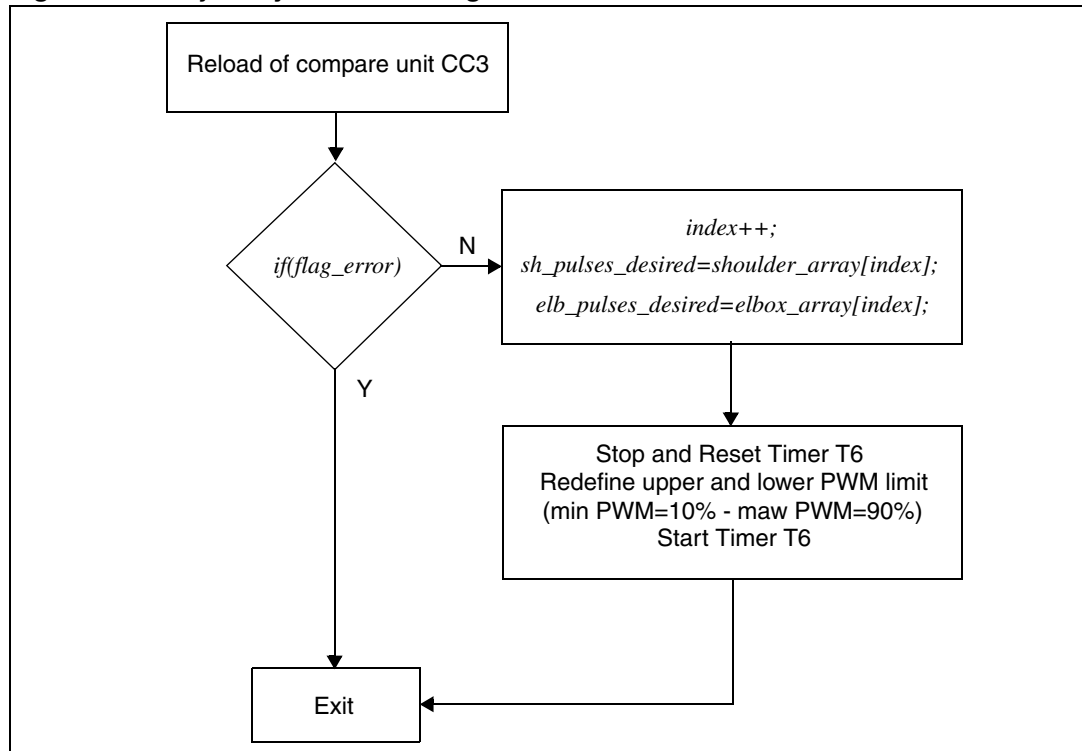


Figure 27. Trajectory table scanning ISR

To avoid oscillations due to external events that could change the motion direction, an additional control algorithm is performed using the ISR associated to timer T6, every 200ms. This implies that if for 10 times no new reference point is given to the control ISR, in the ISR associated to timer T6 the values of PWM are strongly limited until the stop point is reached, drastically reducing the oscillations.

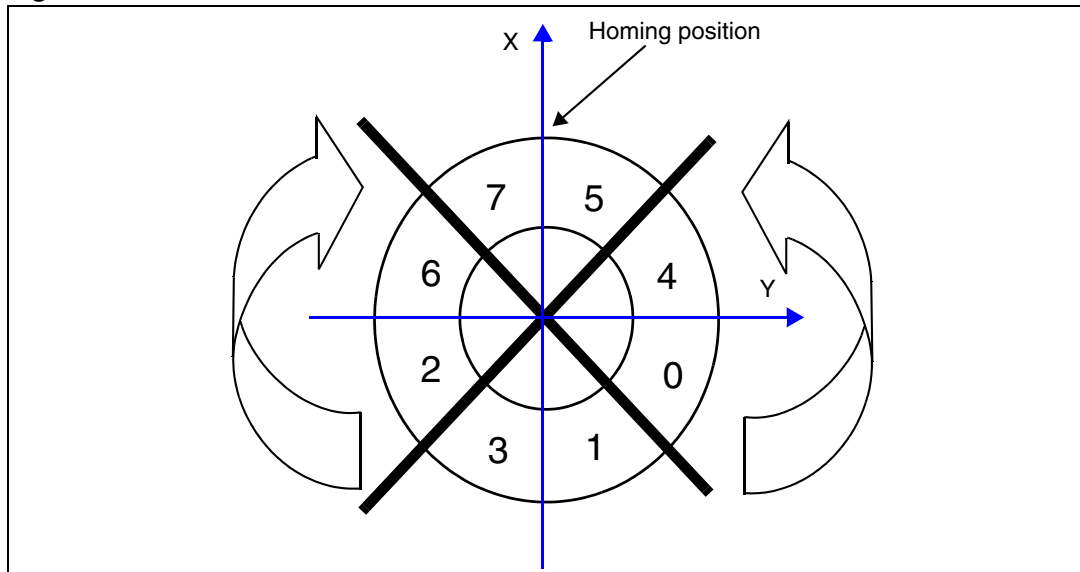
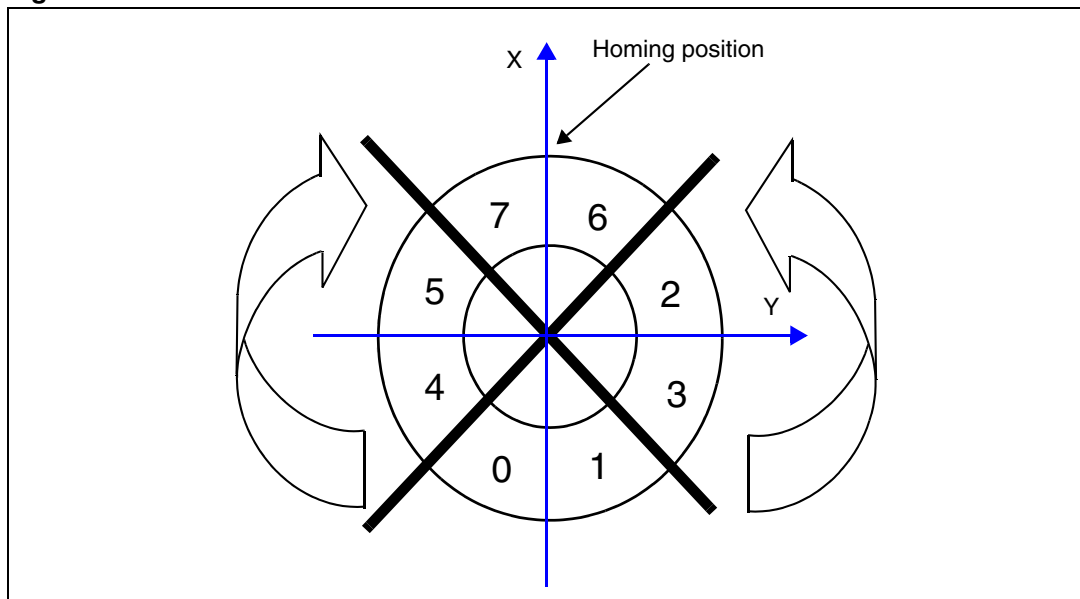
4.3 Homing procedure

The Homing procedure is the most important task a manipulator must execute at start-up. It can be seen as a calibration and alignment phase to the end of which, the manipulator is in a configuration that allows it to perform the other tasks correctly.

For the wafer handler there are many possible home alignment configurations according to the position in which it is placed and to the tasks that it must execute.

For our purpose a homing position has been chosen in which the controlled arm is completely extended. This configuration corresponds to an end effector position at the point of coordinates (481,0).

The homing procedure has been performed through the absolute encoder feedback of the two links, using three GPIOs (one for each bit of the absolute encoder) and one Capture pin (to capture every transition in each bit), for each link. The homing procedure is performed first for the shoulder and then for the elbow. The figures below show the value of the absolute encoder for the shoulder and for the elbow.

Figure 28. Values of the absolute encoder for the shoulder**Figure 29. Values of the absolute encoder for the elbow**

Values in the different sectors represent the decimal value of the bits $b_2b_1b_0$ of the absolute encoder.

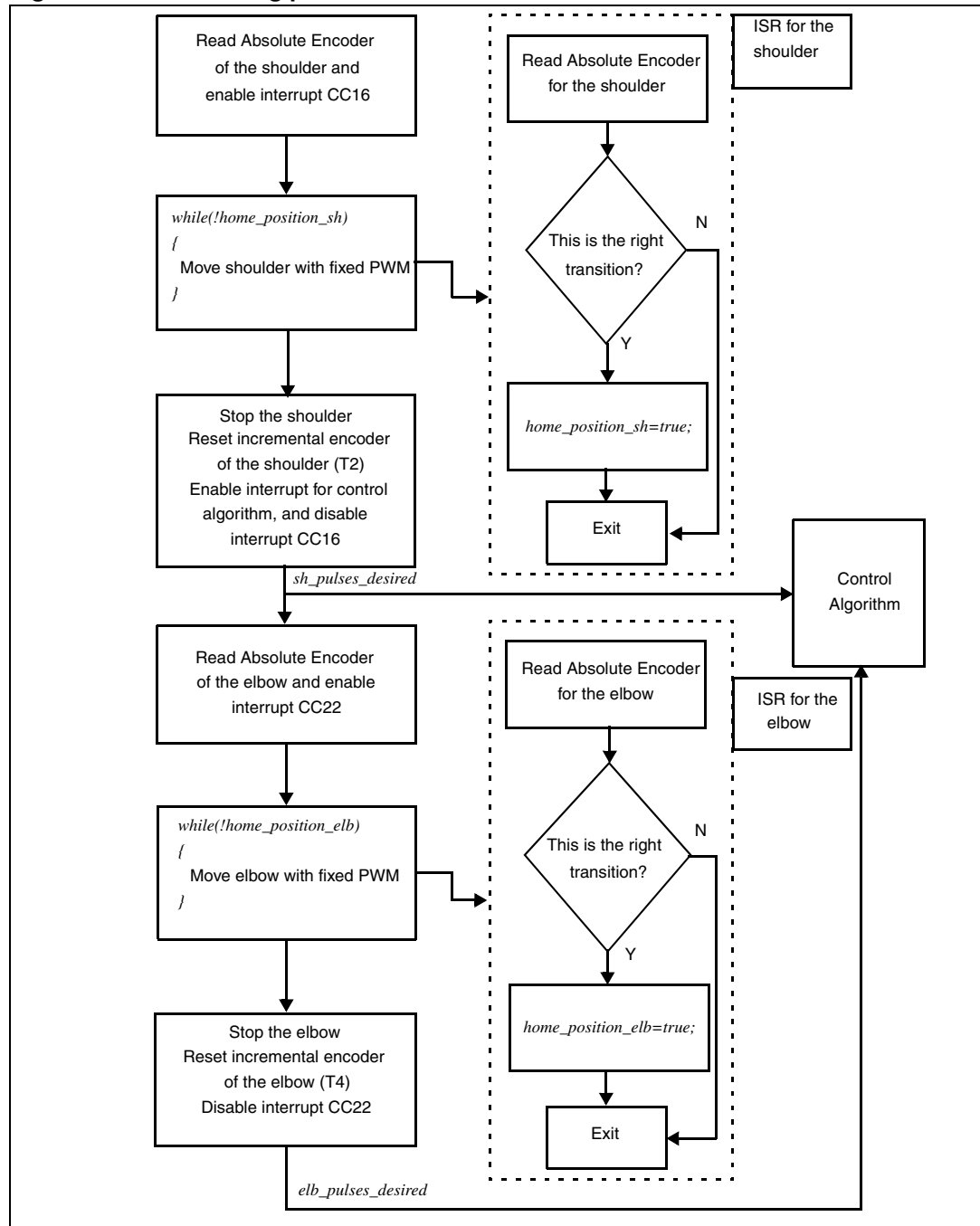
As we can see, the homing position for the shoulder is characterized by a transition between the sectors with values 7 and 5, while the elbow by a transition between the sectors with values 7 and 6.

At the startup of the homing procedure the absolute encoder of the shoulder is read and the ISR associated to the Capture unit CC16 is enabled. Accordingly to the sector in which the shoulder is positioned, the shoulder is moved with a fixed PWM. Every time there is a transition, in the ISR the absolute encoder is read. If the transition is right, then the motor of the shoulder is stopped (PWM=50%) and the timer T2 of the incremental encoder is reset. This value is then passed to the control algorithm.

Now the shoulder is controlled in its home position, the same procedure is executed for the elbow, using the Capture unit CC22. The control of the shoulder ensures that during the home procedure for the elbow there's no movement of the shoulder.

The scheme below shows the homing procedure.

Figure 30. The homing procedure

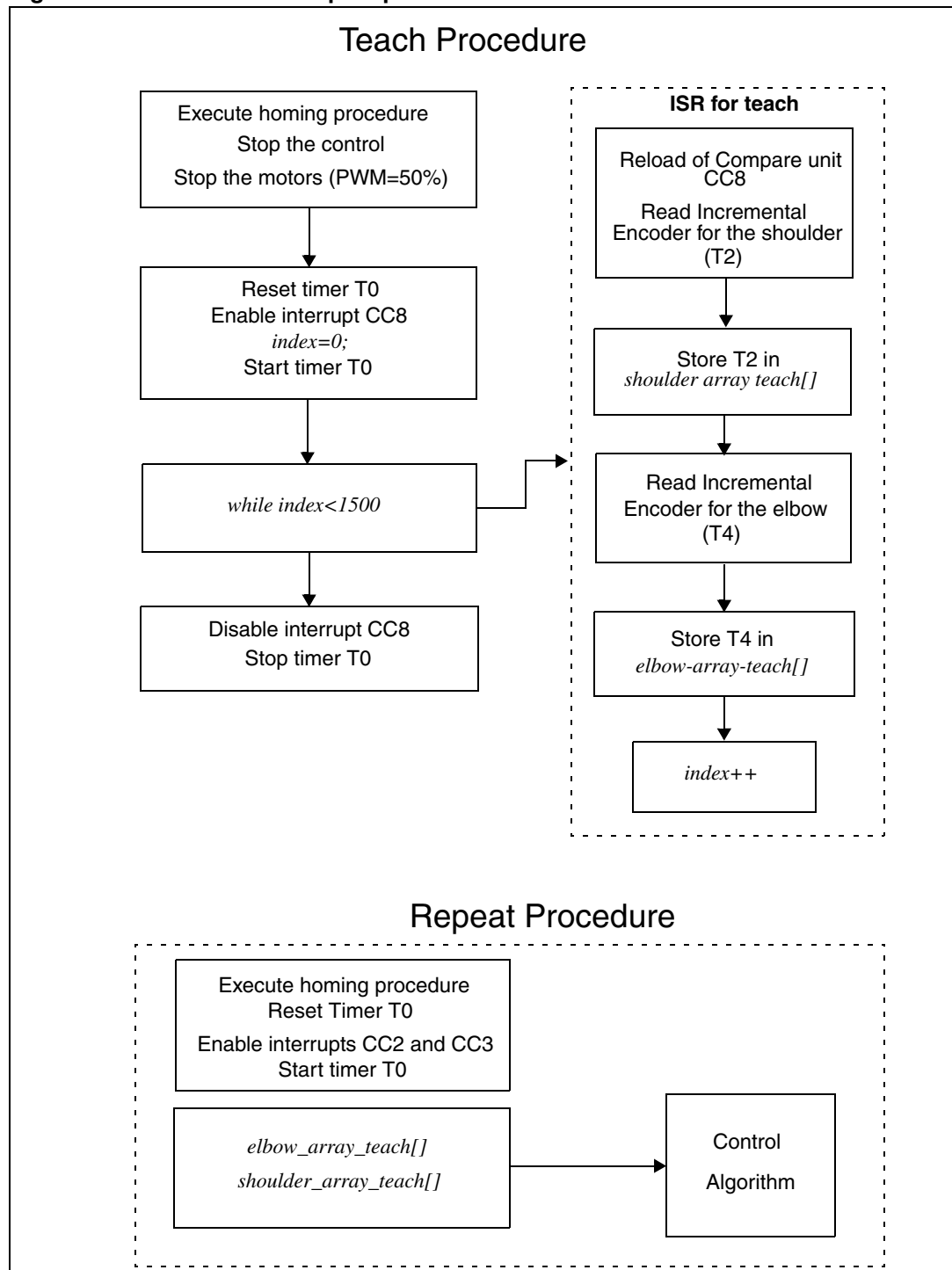


4.4 Teach and Repeat procedure

Robots are mainly programmed using a technique known as *teach and repeat*. Using this technique the control system stores internal joint poses specified by a human operator and then recalls them in sequence to execute a task. In this way it is possible to store complex trajectories allowing the robot can execute them with a good reliability and repeatability.

During the teach phase the control of the manipulator is disabled and the human operator can move the arm as he wants. With a periodic task of 20 ms (associated to the Compare Unit CC8 and the timer T0) the value of the incremental encoders of the two joints (the shoulder and the elbow) are stored in the two corresponding arrays: *shoulder_array_teach[]* and *elbow_array_teach[]*. The teach phase duration depends on the size of the two arrays. Using 3 KByte of the XRAM2 for each array, it is possible to store a trajectory of 30 seconds. In the repeat phase the couple of points of the two arrays are passed to the control algorithm as explained before, as well the scanning of the two arrays. The scheme for teach and repeat procedure is shown in the figure below.

Figure 31. The teach and repeat procedure



5 Revision history

Table 5. Document revision history

Date	Revision	Changes
30-Nov-2006	1	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2006 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com