ESCOLA POLITÈCNICA SUPERIOR D'ENGINYERIA DE VILANOVA
I LA GELTRÚ (EPSEVG)

Av. Victor Balaguer s/n
08800 Vilanova i la Geltru

Tipo:

Proyecto final de carrera

Título:

Programación de un proceso de paletización y posicionamiento flexible en un robot Fanuc.

Iván Navarro Sanz
Tutor: Luis Miguel Muñoz

Tabla de contenidos

1 OBJETO DEL PROYECTO

El proyecto a desarrollar trata de dar una solución automatizada, flexible y precisa un proceso elaborado de una forma completamente manual.

En dicho proceso, se realiza la colocación y pegado de un elemento sobre una pieza de fibra de carbono; en el cual, podemos distinguir de una forma diferenciada tres partes dentro del mismo proceso: el lijado del punto, la limpieza con acetona y el encolado y colocación del elemento sobre la pieza.

Aún siendo un proceso que precisa de gran flexibilidad para solventar los posibles cambios, no deja de ser un trabajo donde es necesaria una gran repetibilidad.

Como producto comercial, donde destaque la repetibilidad, encontramos las aplicaciones robóticas, que, aunque su flexibilidad en programación sea más limitada que la de un PLC, consta con la suficiente flexibilidad para el desarrollo de este proyecto; además de aportar la robustez de su entorno físico que nos será favorable para solucionar los problemas de accesibilidad que nos podamos encontrar.

1.2 Objetivos

El objetivo que se pretende es realizar la programación de un robot, que permita la rápida adición de un punto de aplicación de una forma fácil y sencilla; simplemente añadiéndolo a una "base de datos" de puntos de aplicación y realice las tres partes del proceso, lijado, limpieza y fijado, de una forma altamente flexible y precisa.

2 ANTECEDENTES

Teóricamente el uso de sistemas robóticos podría extenderse a casi todas las áreas imaginables en donde se necesite de la ejecución de tareas mecánicas, tareas hoy ejecutadas por el hombre o imposibles de ejecutar por él (por ej. una exploración sobre el terreno de la superficie marciana). Se entiende, en este contexto, que tarea mecánica es toda actividad que involucra presencia física y movimiento por parte de su ejecutor.

Pero al situarnos en el contexto real, en la práctica, nos damos cuenta de que existen factores que limitan el uso de los robots.

La automatización industrial es el área más relevante y de interés para nosotros. Corresponde al uso de robots en la industria a fin de mejorar, agilizar y aumentar la producción en los diferentes procesos.

Las aplicaciones de los sistemas robóticos podrían ser innumerables. Pero existen dos factores, fuertes y decisivos, que inhiben el crecimiento y desarrollo de esta tecnología. Estos a considerar son limitaciones económicas y limitaciones tecnológicas.

2.2 Antecedentes de la industria aeronáutica

Antes de la producción industrial en serie, la calidad de los productos estaba íntimamente relacionada a la habilidad del artesano. Este tenía su propio sistema de producción, estaba organizado en una unidad en la que no existía una división del trabajo. Desde el diseño, pasando por la adquisición de las materias primas ,la confección de las herramientas y el dominio de las técnicas de esas herramientas hasta la calidad del producto que se manufacturaba y la misma venta, estaban centradas en una misma persona cuanto mas en una organización única que tenia una forma indivisa de trabajo. Se caracterizaban por las siguientes circunstancias:

- Los costos eran altos y no se determinaban en detalle, ni se predecían.
- Las operaciones no se preparaban, ni se analizaban y se confiaba en la capacidad natural de los capataces.
- Los tiempos de producción no se estudiaban, estimándose los plazos de entrega. La programación se reducía a las anotaciones que llevaban los capataces en sus libretas.
- El nivel de producción era reducido.
- Las herramientas eran simples y servían como auxiliares del obrero y normalmente de propiedad de los obreros.

- El aprendizaje del trabajo era por repetición.

Después de la revolución industrial se observaron las siguientes tendencias en las empresas:

- A la aplicación de métodos científicos de organización y dirección.
- Al crecimiento de las mismas.
- A la especialización.
- A la normalización.
- A la extrema división del trabajo.

2.2 Fig. 1 Fotografía de un avión mono hélice.

La necesidad de elevar la eficiencia de los sistemas productivos industriales llevo a efectuar una división del trabajo dentro de lo que se fueron constituyendo en empresas cada vez de mayor tamaño y con una tendencia a dividir el trabajo en diferentes especialidades.

Esta división del trabajo produjo beneficios con respecto al sistema anterior, pero se perdió la unidad conceptual que tenía el artesano con respecto al concepto de calidad de su producto.

Con la división del trabajo dentro de la empresa y la especialización de las distintas áreas, el producto comienza a ser la resultante de ideas concebidas en un sector, de medios de producción concebidos en otro, de especificaciones para producir originadas en un tercero, de elementos comprados bajo la responsabilidad de otro sector más y como consecuencia

de esto, la salida de un producto es el resultado de actividades fragmentadas en cuanto a su responsabilidad y ejecución.

La industria aeronáutica ha ido cambiando conforme la tecnología va evolucionando, los datos aeronáuticos se han convertido en un componente crucial de los sistemas y de seguridad operacional, ya que los estados deben asegurar que se cuenta con información aeronáutica de alta calidad.

La tecnología como recurso estratégico y su consiguiente adopción a través de los procesos de automatización es costosa, y el tratar de adoptarla tal y como funciona en el sector automovilístico genera el problema de invertir cuantiosas sumas de dinero para proveerse de los mecanismos que permitan obtener beneficios tangibles.

2.2 Fig. 2 Avión de combate a reacción

2.1.1 Motores Aeronáuticos

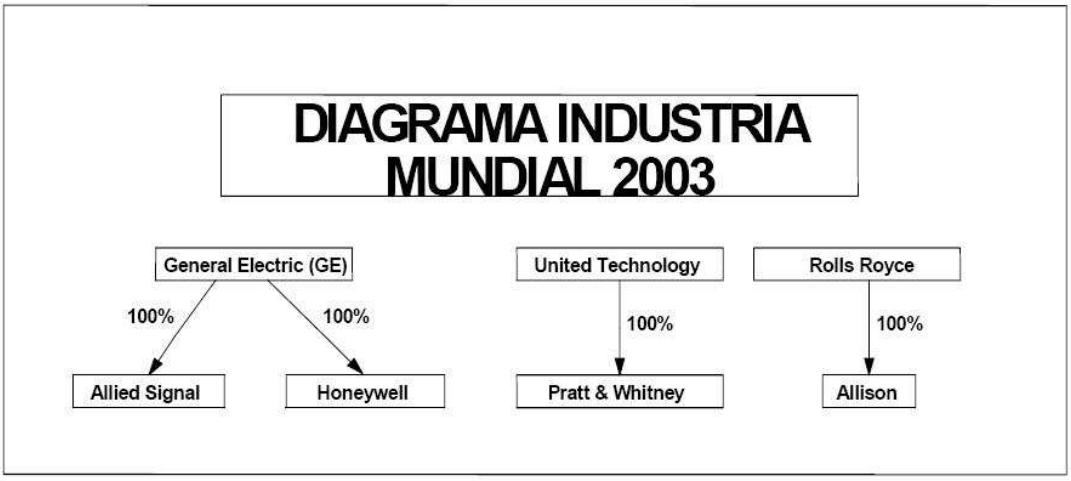
La industria de motores aeronáuticos, a nivel mundial, está, fundamentalmente, representada por tres grandes compañías, véase 2.1.1 Fig. 1, todas de carácter e implantación multinacional pero de origen anglosajón:

- General Electric, G.E., en USA.

- Pratt and Whitney, P&W, (División de United Technologies) en Estados

Unidos y Canadá.

- Rolls-Royce, R&R, en U.K.



2.1.1 Fig 1 Principales fabricantes de motores.

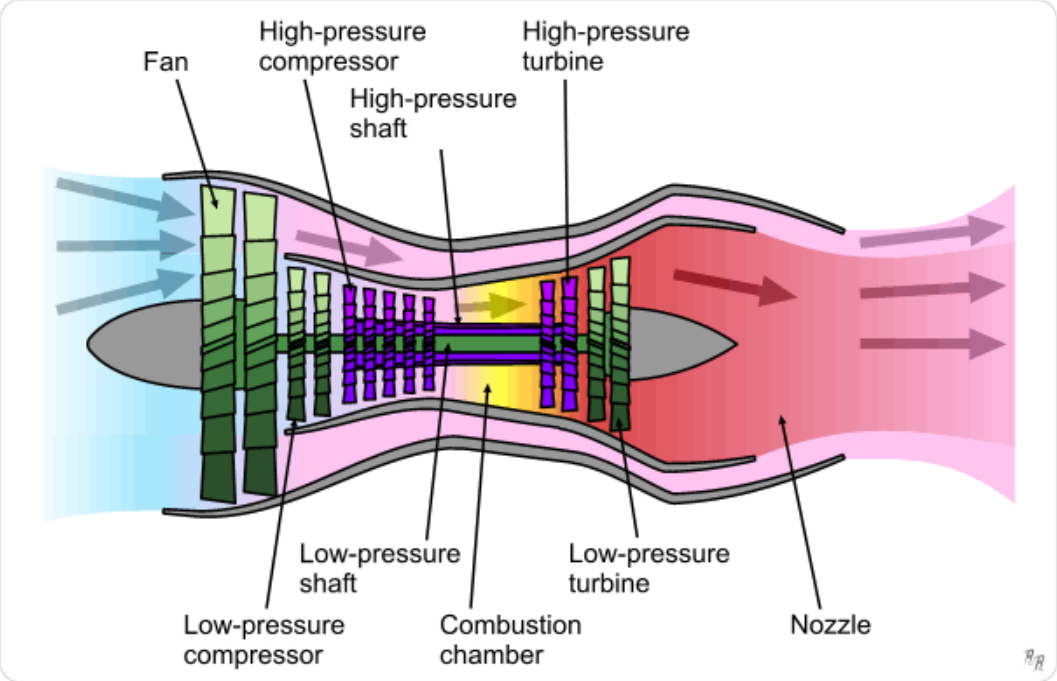
Cada una de estas tres con productos propios en casi toda la gama de empujes (excepto los muy pequeños) y también con productos propios en los motores de aplicación militar. Fuera de estos tres grandes fabricantes tanto en Estados Unidos como en Europa operan otros fabricantes en gamas de menor empuje así como en turbohélices y turboejes (BMW-RR, Turbomeca, Allison o Allied Signal entre otros).

Los motores aeronáuticos se dividen en tres grandes categorías: motores alternativos, motores a reacción y motores cohete.

Hay tres tipos básicos de motores de reacción: el turborreactor, el turbohélice (dentro de este grupo incluimos al turboeje, que es el motor empleado en helicópteros y que no genera prácticamente empuje por chorro, sólo genera tracción) y el turbofan.

El turbofan es utilizado por la mayoría de los aviones comerciales de transporte de pasajeros que vuelan a velocidades transónicas y alcanzan números de Mach 0.8.

Este tipo de motor, incluye una gran hélice interna (fan) y dos corrientes de aire que fluyen a través del motor. La corriente principal viaja a través de todos los componentes como sucede en un turborreactor, Es decir, pasa por la cámara de combustión, mientras que la corriente secundaria generalmente es impulsada a través de una tobera de eyección para mezclarse después, o no, con la corriente primaria de escape; tal y como podemos observar en la imagen 2.1.1 Fig. 2.



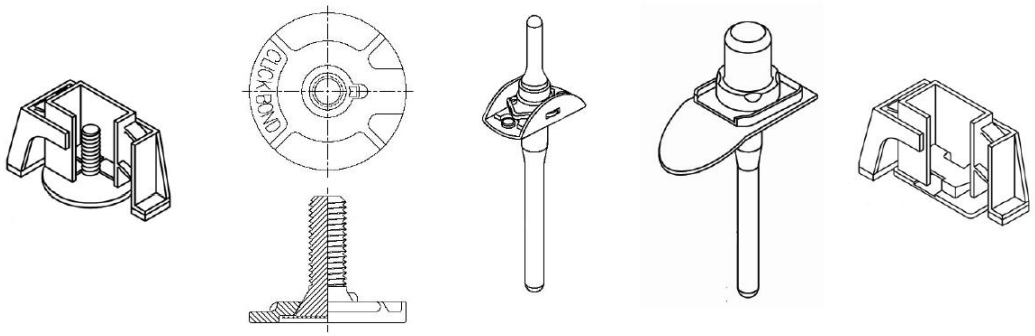
2.1.1 Fig 2. Esquema de funcionamiento de un motor turbofan

En nuestra aplicación manipularemos una parte del carenado de la turbina, colocándole unos fasteners (click-bonds) que sirvan de sujeción a una tela asfáltica que tendrá la función de aislar dicho carenado de las altas temperaturas alcanzadas por el motor.

2.1.2 Sistema de sujeción o fastener

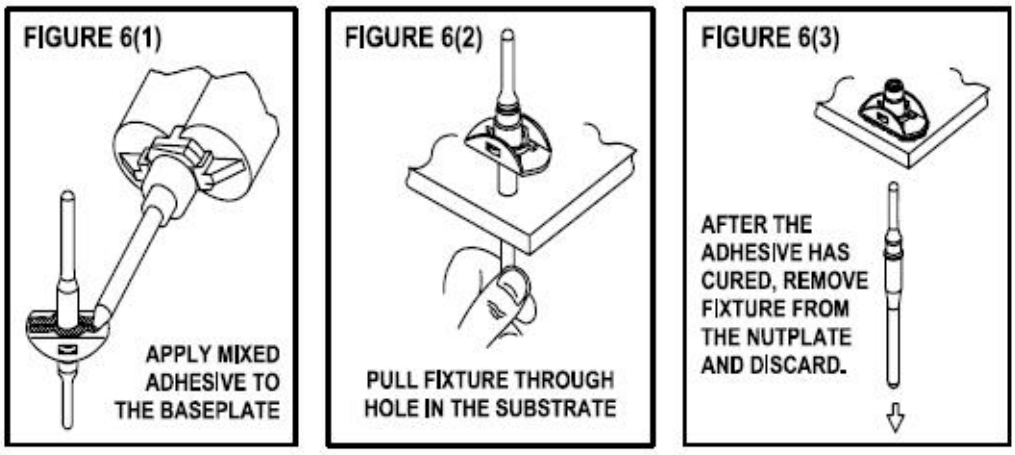
El sistema de sujeción que se adherirá a la pieza, es un sistema de sujeción mediante pegado, líder en la industria aeronáutica de la marca Click-bond. En la figura 2.1.2 Fig. 1 podemos ver diferentes tipos de sujeciones de la marca Click-bond.

Las sujeciones o Fasteners, están especialmente diseñados para ser fijados a diversas clases de estructuras mediante el uso del pegamento. La línea de sujeciones incluye varillas roscadas, remaches, grapas para la fijación de cables, pilares, sistemas de montaje de aislamiento, etc.



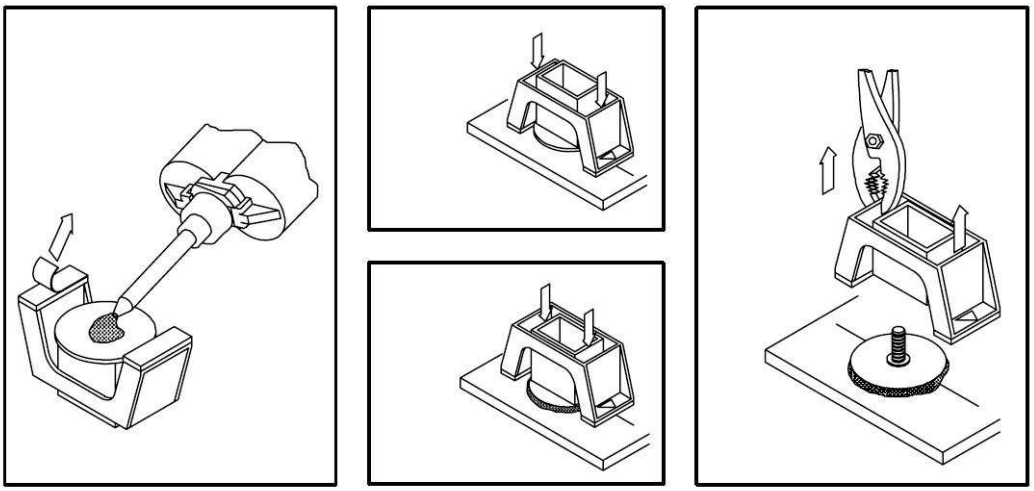
2.1.2 Fig 1. Diferentes tipos de fasteners

Todas las sujeciones de Click-Bond utilizan un proceso de self-fixturing único para mantener la posición exacta y asegurar una presión constante, como muestran las imágenes 2.1.2 Fig 2 y Fig. 3, sobre el material al que va a ser fijado, durante el tiempo de secado del pegamento. Este elemento puede ser externo o interno.



2.1.2 Fig. 2 Modo de uso I

Desde 1981, los Click-bond han proporcionado una variedad de sujeciones a la industria aeroespacial, marítima, militar e industrias de transporte. Se ha trabajado con fabricantes principales para desarrollar una tecnología de fijación consolidada para aviones de pasajeros, aviones militares de transporte, helicópteros, vehículos de lanzamiento, satélites, yates, autobuses, camiones, coches de carreras y artículos de deporte.



2.1.2 Fig. 3 Modo de uso II

El proceso de instalación actualmente, se realiza de una forma completamente manual.

- Paso1: Se aplica el adhesivo en la base del fastener y se retiran las protecciones de los pads.
- Paso 2: se coloca el Click Bond en la superficie deseada, y se presiona la carcasa hasta que los pads se queden adheridos.
- Paso 3: se presiona la parte central de la carcasa donde va embarcado el fastener, para que el clic bond ejerza una presión constante sobre la pieza hasta que el adhesivo se haya secado.
- Paso 4: Una vez el adhesivo se ha solidificado, se procede a retirar la carcasa, dejando el fastener pegado en la posición deseada.

2.2 Robótica y automatización

Son disciplinas surgidas en diferentes épocas. La robótica nace en décadas recientes para complementarse con la automatización, aportándole como elemento innovador cierto grado de inteligencia.

En el contexto industrial, la automatización es como una tecnología que está relacionada con el empleo de sistemas mecánicos, electrónicos y basados en la informática en la operación y control de la producción. Este concepto, para ser actualizado, debe incluir el uso de robots.

2.2 Fig1. Robot de manipulación

El robot industrial forma parte del progresivo desarrollo de la automatización industrial, favorecido notablemente por el avance de las técnicas de control por computadora, y contribuye de manera decisiva a la automatización en los procesos de fabricación de series de mediana y pequeña escala.

2.2.1 Robótica Industrial

Se entiende por Robot Industrial a un dispositivo de maniobra destinado a ser utilizado en la industria y dotado de uno o varios brazos, fácilmente programable para cumplir operaciones diversas con varios grados de libertad y destinado a sustituir la actividad física del hombre en las tareas repetitivas, monótonas, desagradables o peligrosas.

Realmente, los Robots no incorporan nada nuevo a la tecnología en general, la novedad radica en la particularidad de su arquitectura y en los objetivos que se procura con los mismos. El trabajo del Robot se limita generalmente a pocos movimientos repetitivos de sus ejes, estos son casi siempre 3 para el cuerpo y 3 para la mano o puño, su radio de

acción queda determinado por un sector circular en el espacio donde este alcanza a actuar. Cuando las partes o piezas a manipular son idénticas entre sí y se presentan en la misma posición, los movimientos destinados a reubicar o montar partes se efectúan mediante dispositivos articulados que a menudo finalizan con pinzas.

La Federación Internacional de la Robótica (IFR) estableció en 1998 una clasificación de las aplicaciones de la Robótica en el sector manufacturero:

-Manipulación en fundición

- Moldes
- Otros

-Manipulación en moldeado de plásticos

-Manipulación en tratamientos térmicos

-Manipulación en la forja y estampación

-Soldadura.

- Por arco
- Por puntos
- Por gas
- Por láser
- Otros

-Aplicación de materiales

- Pintura
- Adhesivos y secantes
- Otros

-Mecanización

- Carga y descarga de máquinas
- Corte mecánico, rectificado, desbardado y pulido
- Otros

-Otros procesos

Láser

Chorro de agua

Otros

-Montaje.

Montaje mecánico

Inserción

Unión por adhesivos

Unión por soldadura

Manipulación para montaje (pick and place)

Otros

Paletización

Medición, inspección, control de calidad

Manipulación de materiales

Formación, enseñanza e investigación

2.2 Fig. 2. Célula robotizada.

Esta clasificación pretende englobar la mayor parte de los procesos robotizados en la actualidad aunque, como se ha indicado anteriormente, se pueden encontrar aplicaciones particulares que no aparecen de manera explícita en esta clasificación.

2.2.2 Manipulación o Pick and place.

La misión de un robot trabajando en un proceso de pick and place consiste en recoger piezas de un lugar y depositarlas en otro, como se refleja en la imagen 2.2.2 Fig. 1. La complejidad de este proceso puede ser muy variable, desde el caso más sencillo en el que el robot recoge y deja las piezas en una posición prefijada, hasta aquellas aplicaciones en las que el robot precise de sensores externos, como visión artificial o tacto, para determinar la posición de recogida y colocación de las piezas.

2.2.2 Fig. 1 Robot para aplicación Pick and place.

Al contrario que en las operaciones de paletizado, las tareas de picking suelen realizarse con piezas pequeñas (peso inferior a 5Kg) necesitándose velocidad y precisión.

2.2.3 Paletización.

La paletización es un proceso básicamente de manipulación, consistente en disponer de piezas sobre una plataforma o bandeja (palet). Las piezas en un palet ocupan normalmente posiciones predeterminadas, procurando asegurar la estabilidad, facilitar su manipulación y optimizar su extensión. Los palets son transportados por diferentes sistemas (cintas transportadoras, carretillas, etc.) llevando su carga de piezas, bien a lo largo del proceso de fabricación, bien hasta el almacén o punto de expedición.

Dependiendo de la aplicación concreta, un palet puede transportar piezas idénticas (para almacenamiento por lotes por ejemplo), conjuntos de piezas diferentes, pero siempre los mismos subconjuntos procedentes de ensamblados) o cargas de piezas diferentes y de composición aleatoria (formación de pedidos en un almacén de distribución).

Existen diferentes tipos de maquinas especificas para realizar operaciones de paletizado. Estas frente al robot, presentan ventajas en cuanto a velocidad y coste, sin embargo, son rígidos en cuanto a su funcionamiento, siendo incapaces de modificar su tarea de carga y descarga. Así mismo, actualmente la diferencia de coste es aproximadamente igual, y los

robots realizan con ventaja aplicaciones de paletización en las que la forma, numero o características generales de los productos a manipular, cambian con relativa frecuencia. En estos casos, un programa de control adecuado permite resolver la operación de carga y descarga, optimizando los movimientos del robot, aprovechando la capacidad del palet o atendiendo a cualquier otro imperativo. Por otro lado, el robot industrial siempre se puede reutilizar para cualquier operación industrial debido a su flexibilidad.

2.2.3 Fig 1. Célula de palletizado

Generalmente, las tareas de paletización implican el manejo de grandes cargas, de peso y dimensiones elevadas. Por este motivo, los robots empleados en este tipo de aplicaciones acostumbran a ser robots de gran tamaño, con una capacidad de carga de 10 a 150 kg aproximadamente. En la imagen 2.2.3 Fig 1 podemos ver una celula de paletizado donde el robot ha de mover una gran cantidad de sacos de un peso considerable.

No obstante, se pueden encontrar aplicaciones de paletización de pequeñas piezas, en las que un robot con una capacidad de carga de 5Kg. es suficiente.

Las denominadas tareas de Pick and place, aunque en general con características diferentes al paletizado, guardan estrecha relación con este.

2.2.3 Fig. 2 Célula de depaletizado

El lenguaje V+ posee diferentes facilidades para modelar determinadas tareas, como la definición de palets (pallet.frame) a partir de tres localizaciones especificadas, o para el seguimiento y manipulación de objetos en cintas transportadoras, que se mueven a velocidad constante (definiendo unas ventanas de trabajo sobre la propia cinta).

Un ejemplo típico de aplicación de robot al paletizado seria la formación de palets de cajas de productos alimenticios procedentes de una línea de empaquetado. En estos casos, cajas de diferentes productos llegan aleatoriamente al campo de acción del robot. Ahí son identificadas bien por una célula de carga, por alguna de sus dimensiones, o por un código de barras. Conocida la identidad de la caja, el robot procede a recogerla y a colocarla en uno de los diferentes palets que, de manera simultanea, se están formando.

El propio robot gestiona las líneas de alimentación de las cajas y de palets, a la vez que toma las decisiones necesarias para situar la caja en el palet con la posición y orientación adecuadas de una manera flexible.

El robot podrá ir equipado con una serie de ventosas de vacío y su capacidad de carga estaría entorno a los 50 kg aproximadamente.

3 PROGRAMACIÓN DE ROBOTS

Para cualquier maquina controlada por un sistema informático, se necesita un medio con el que poder gobernar su funcionamiento; este medio es el *lenguaje de programación*. En un robot industrial, el rendimiento obtenido, viene determinado por la adaptación a la tarea a realizar y la sencillez de manejo de dicho lenguaje.

Existen diversas maneras de comunicarse con un robot; de todas ellas, la enseñanza y repetición, o guiado, es la solución más utilizada para los robots industriales. Este método implica enseñar al robot dirigiéndole los movimientos que el programador desea que realice.

La enseñanza y repetición se lleva a cabo normalmente con los siguientes pasos:

1) dirigiendo al robot con un movimiento lento utilizando el control manual para realizar la tarea completa y grabando los ángulos del movimiento del robot en los lugares adecuados para que vuelva a repetir el movimiento;

2) reproduciendo y repitiendo el movimiento enseñado;

3) si el movimiento enseñado es correcto, entonces se hace funcionar al robot a la velocidad correcta en el modo repetitivo.

Los lenguajes de programación de alto nivel suministran una solución más general para resolver el problema de comunicación. Durante años, los robots han sido utilizados con éxito en áreas tales como soldadura por arco voltaico o pintura con spray mediante la técnica del guiado.

3.2 Clasificación de la programación usada en robótica.

Podemos clasificar la programación empleada en robótica según la responsabilidad del operador sobre las acciones de control o según la toma de decisiones del sistema basada en la modelación del mundo exterior, cuando se describe la tarea y el entorno

La programación explícita es la utilizada en las aplicaciones industriales y consta de dos técnicas fundamentales; programación Gestual y Programación Textual.

Si realizamos la programación del robot guiando el brazo del robot y realizando el grabado de la trayectoria deseada, estamos hablando de programación gestual; este tipo de programación, .exige el empleo del manipulador en la fase de enseñanza, o sea, trabaja "on-line". Sin embargo. En la programación textual, no necesitamos de la participación del brazo, debido a que todas las acciones a realizar se especifican mediante el uso de un lenguaje de alto nivel.

3.1.1 Programación gestual o directa

Podemos dividir la programación gestual en dos clases; aprendizaje directo o mediante un dispositivo de enseñanza.

En el aprendizaje directo, el operario desplaza la muñeca del manipulador o del brazo maestro, realizando la trayectoria a seguir por el brazo. Esta técnica de aprendizaje está muy extendida en los procesos de pintura, donde un operario con escasos conocimientos de programación pero con experiencia en el trabajo puede realizar el memorizado de los tramos a recorrer y aquellos puntos en los que la pistola debe expulsar una cierta cantidad de pintura.

Esta clase de programación tiene pocas posibilidades de edición ya que para generar una trayectoria continua, es preciso definir una gran cantidad de puntos; cuya reducción origina discontinuidades.

En el caso de la programación mediante un dispositivo de enseñanza, determinamos las acciones y movimientos del brazo a través de un elemento especial que suele estar constituido por botones, teclas, joystick…

Hay que tener en cuenta que los dispositivos de enseñanza modernos no sólo permiten controlar los movimientos de las articulaciones del manipulador, sino que pueden, también, generar funciones auxiliares, como:

- Selección de velocidades

- Generación de retardos

- Señalización del estado de los sensores

- Borrado y modificación de los puntos de trabajo

- Funciones especiales

Estos lenguajes se utilizan con el robot "in situ", recordando las normas de funcionamiento de un video doméstico, ya que disponen de unas instrucciones similares: PLAY (reproducir), RECORD (grabar), FF (adelantar), FR (atrasar), PAUSE, STOP, etc. Además, puede disponer de instrucciones auxiliares, como INSERT (insertar un punto o una operación de trabajo) y DELETE (borrar).

Con este tipo de programación tampoco es necesario conocer ningún tipo de lenguaje de programación, el operario solamente deberá habituarse al empleo del dispositivo de enseñanza.

Los lenguajes de programación gestual, además de necesitar al propio robot en la confección del programa, carecen de adaptabilidad en tiempo real con el entorno y no pueden tratar, con facilidad, interacciones de emergencia.

Los lenguajes más conocidos en programación gestual punto a punto son el FUNKY (creado por IBM), y el T3 (creado por Cincinnati Milacrom).

En el lenguaje FUNKY se utiliza un mando (joystick), que dispone de un comando especial para centrar la pinza sobre el objeto y controlar los movimientos, mientras que el T3 dispone de un dispositivo de enseñanza ("teach pendant").

El procesador usado en T3 es el AMD 2900 ("bit slice"), mientras que en el FUNKY está constituido por el IBM SYSTEM-7. En la programación gestual, al intervenir el brazo en el desarrollo de las acciones a realizar en la aplicación y en el trazado, siendo necesaria la realización del programa de manera “on-line”.

Algunos de los lenguajes, mas importantes, que tratan los movimientos punto a punto son:

ANORAD- Este lenguajes es una transformación de un lenguaje de control numérico de la casa ANORAD CORPORATION, utilizado para el robot ANOMATIC. Utiliza como procesador el microprocesador 68000 de Motorola de 16/32 bits.

EMILY- Es un lenguaje creado por IBM para el control de uno de sus robots. Usa el procesador IBM 370/145 SYSTEM 7 y está escrito en Ensamblador.

MAL- Fue creado en el Politécnico de Milán para el robot SIGMA, con un Mini-multiprocesador. Es un lenguaje del tipo intérprete, escrito en FORTRAN.

RCL- Aplicado al robot PACS y desarrollado por RPI, emplea como CPU, un PDP 11/03. Es del tipo intérprete y está escrito en Ensamblador.

RPL- Esta dotado con un LSI-II como procesador central que se aplica a los robots PUMA. Fue diseñado por SRI INTERNATIONAL.

SIGLA- Fue desarrollado por OLIVETTI para su robot SUPER SIGMA. Emplea un mini-ordenador con 8K de memoria. Escrito en Ensamblador, es de tipo intérprete.

VAL- Fue diseñado por UNIMATION INC para sus robots UNIMATE y PUMA. Utiliza como CPU un LSI-II, que se comunica con procesadores individuales que regulan el servocontrol de cada articulación. Las instrucciones (en inglés) son sencillas e intuitivas.

Estos lenguajes mantienen las características de los movimientos primitivos, ya sea en coordenadas articulares o cartesianas. Tienen como ventajas destacables, los saltos condicionales y subrutinas, además de un aumento de las operaciones con sensores.

Estos lenguajes son de tipo intérprete, con excepción del RPL, que tiene un compilador. La mayoría dispone de comandos de tratamiento a sensores básicos: tacto, fuerza, movimiento, proximidad y presencia.

3.1.2 Programación textual explicita

Los lenguajes de programación textual se encuadran en varios niveles, según se realice la descripción del trabajo del robot. Se relacionan a continuación, en orden creciente de complejidad; lenguajes elementales, que controlan directamente el movimiento de las articulaciones del manipulador, lenguajes dirigidos a posicionar el elemento terminal del manipulador, lenguajes orientados hacia el objeto sobre el que opera el sistema, y lenguajes enfocados a la tarea que realiza el robot.

En la programación textual, no se requiere de la intervención del robot, dicho programa se realiza mediante un texto de instrucciones o sentencias adecuadas. Según las características del lenguaje, pueden confeccionarse programas de trabajo complejos, con inclusión de saltos condicionales, empleo de bases de datos, posibilidad de creación de módulos operativos intercambiables, capacidad de adaptación a las condiciones del mundo exterior, etc.

Dentro de la programación textual, existen dos grandes grupos, de características netamente diferentes; programación textual explícita y programación textual especificativa.

El programa, en la programación textual explicita, consta de una secuencia de órdenes o instrucciones concretas, que van definiendo las operaciones necesarias para llevar a cabo la aplicación. Este tipo de programación engloba a los lenguajes que definen los movimientos punto por punto, similares a los de la programación gestual. Con este tipo de programación la labor del tratamiento de las situaciones anormales, colisiones, etc… queda a cargo del programador. Podemos encontrar dos subgrupos dentro de la programación textual.

1º nivel de movimiento elemental, que comprende los lenguajes dirigidos a controlar los movimientos del brazo manipulador.

2º. Nivel estructurado, que intenta introducir relaciones entre el objeto y el sistema del robot, para que los lenguajes se desarrollen sobre una estructura formal.

Se puede decir que los lenguajes correspondientes a este tipo de programación adoptan la filosofía del PASCAL. Describen objetos y transformaciones con objetos, disponiendo, muchos de ellos, de una estructura de datos arborescente. El uso de lenguajes con programación explícita estructurada aumenta la comprensión del programa, reduce el tiempo de edición y simplifica las acciones encaminadas a la consecución de tareas determinadas.

Algunos de los lenguajes más importantes de programación explícita son:

AL- Proporciona las definiciones necesarias acerca de los movimientos relacionados con los elementos sobre los que trabaja el brazo. AL fue diseñado por el laboratorio de Inteligencia Artificial de la Universidad de Stanford, con estructuras de bloques y de control similares al ALGOL (lenguaje en el que fue escrito). Está dedicado al manipulador de Stanford, utilizando como procesadores centrales, a un PDP 11/45 y un PDP KL-10.

HELP- Fue creado por GENERAL ELECTRIC, para su robot ALLEGRO y está escrito en PASCAL/FORTRAN. Este lenguaje permite el movimiento simultáneo de varios brazos y dispone de un conjunto especial de subrutinas para la ejecución de cualquier tarea. Utiliza como CPU un PDP 11.

MAPLE- Fue escrito por IBM, como intérprete en lenguaje PL-1, para el robot de la misma empresa. El cual, tiene capacidad para soportar informaciones de sensores externos. Utiliza como CPU un IBM 370/145 SYSTEM 7.

PAL- Fue desarrollado por la Universidad de Purdure para el manipulador de Stanford. Esta escrito en FORTRAN y ensamblador, y es un intérprete, capaz de aceptar sensores de fuerza y de visión. Cada una de sus instrucciones para mover el brazo del robot en coordenadas cartesianas, es procesada para que satisfaga la ecuación del procesamiento. Como CPU utiliza un PDP 11/70.

MCL- Fue creado por la compañía MC DONALL DOUGLAS, como ampliación de su lenguaje de control numérico APT. Es un lenguaje compilable, apto para la programación de robots "off-line".

MAL EXTENDIDO- Procede del Politécnico de Milán, al igual que el MAL. Incorpora elementos de programación estructurada y se utiliza en el robot SIGMA.

3.1.3 Programación textual especificativa

En la programación textual especificativa, el usuario o programador, describe las especificaciones de los productos y las tareas a realizar, mediante una modelización. Se debe introducir en una base de datos, más o menos compleja, el modelo del universo que rodea al robot, que suelen ser de tipo geométrico y no físico. Suele requerir computadoras potentes para procesar una abundante información.

La creación de lenguajes de muy alto nivel transferirá una gran parte del trabajo de programación, desde el usuario hasta el sistema informático; éste resolverá la mayoría de los problemas, combinando la Automática y la Inteligencia Artificial.

De los lenguajes de programación explícita a nivel de objeto destacamos los tres más interesantes:

RAPT- El lenguaje RAPT fue creado en la Universidad de Edimburgo, departamento de Inteligencia Artificial; está orientado, en especial, al ensamblaje de piezas. Destinado al robot FREDY, utiliza, como procesador central, a un PDP 10.

Es un intérprete y está escrito en lenguaje APT.

Este lenguaje se basa en definir una serie de planos, cilindros y esferas, que dan lugar a otros cuerpos. Para modelar un cuerpo, se confecciona una biblioteca con sus rasgos más representativos. Seguidamente, se define los movimientos que ligan a los cuerpos a ensamblar (alinear planos, encajar cilindros, etc.).

AUTOPASS- Fue creado por IBM para el ensamblaje de piezas. Utiliza instrucciones, muy comunes, en inglés. Precisa de un ordenador de varios Megabytes de capacidad de memoria.

Además de indicar, como el RAPT, puntos específicos, prevé, también, colisiones y genera acciones a partir de las situaciones reales.

El AUTOPASS realiza todos sus cálculos sobre una base de datos, que define a los objetos como poliedros de un máximo de 20,000 caras. Está escrito en PL/1 y es intérprete y compilable.

LAMA- Fue creado por el laboratorio de Inteligencia Artificial del MIT, para el robot SILVER, en el que se orientaron hacia el ajuste de conjuntos mecánicos. Aporta más inteligencia que el AUTOPASS y permite una buena adaptación al entorno.

La operatividad del LAMA se basa en tres funciones principales:

- Creación de la función de trabajo. Operación inteligente.

- Generación de la función de manipulación.

- Interpretación y desarrollo, de forma interactiva, de una estrategia de realimentación para la adaptación al entorno de trabajo.

3.2 Programación de un robot FANUC

La programación de una aplicación de un robot, consiste en una sucesión de comandos y órdenes dadas por el usuario o programador para realizar una operación concreta.

Además, la información del programa, describe como debe el robot realizar dicha operación. Un programa, contiene información detallada que define los atributos del programa.

Esta información detallada consiste en:

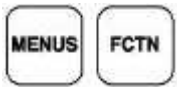
- Fecha de creación del programa, de modificación, presencia o ausencia de datos de movimiento y tamaño del programa.
- Información relacionada con la ejecución, como el nombre del programa, el subtipo, comentarios, mascara de grupo, la protección de grabado o la interrupción de inhabilitación.

3.2.1 Descripción del Teach Pendant

3.2.1 Fig. 1 Teclas de función y representación del Teach Pendant

Las teclas de función (F) para seleccionar un menú de función en la última línea de la pantalla.

Para habilitar más tecla de función en la página siguiente.



La tecla MENUS para visualizar el menú de pantalla.

La tecla FCTN para visualizar el menú de función.

La tecla SELECT para visualizar la pantalla de selección del programa.

La tecla EDIT para visualizar la pantalla de edición del programa.

La tecla DATA para visualizar la pantalla de datos del programa.

La tecla MAN FCTNS visualiza la pantalla de operación manual.

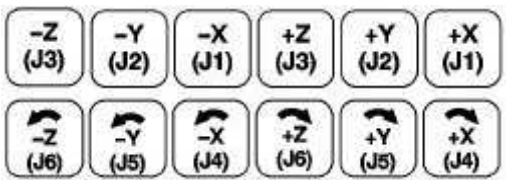
La tecla STATUS visualiza la pantalla de posición actual.

La tecla E/S visualiza la pantalla de E/S.

La tecla POSN visualiza la pantalla de posición actual

La tecla SHIFT se utiliza para la habilitación de movimiento del robot, programar los datos de posición, y arrancar un programa.

Las teclas Shift derecha e izquierda tienen la misma función.



Las teclas de movimiento son efectivas mientras se mantiene pulsada una tecla Shift. Se utilizan para la habilitación de movimiento.

La tecla COORD selecciona un sistema de coordenadas de movimiento manual. Cada vez que se pulsa la tecla COORD, selecciona el siguiente tipo de movimiento en el orden: JOINT, JGFRM, World frame, TOOL, USER.

Cuando se pulsa esta tecla mientras se mantiene pulsada una tecla Shift, aparece un menú de movimiento para el cambio del sistema de coordenadas.

La tecla variación de velocidad. Cada vez que se pulsa varia en el orden: VFINE, FINE, 1%, 5%, 50%, 100%. (Cambio del 5% de la cantidad para el 5% o menos y cambio del 5% de la cantidad para el 5% o más.

La tecla FWD o la tecla BWD (+tecla SHIFT)arranca un programa. Cuando se libera la tecla shift durante la regeneración, se interrumpe el programa.

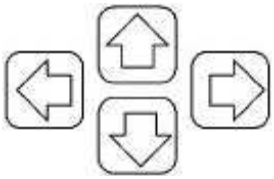
La tecla HOLD provoca que se interrumpa un programa.

La tecla STEP selecciona paso a paso o operación en ciclo continuo.

La tecla PREV vuelve a almacenar el estado más reciente. En algunos casos, la tecla no puede volver inmediatamente al estado anterior.

La tecla ENTER introduce, valida y selecciona un número o un menú.

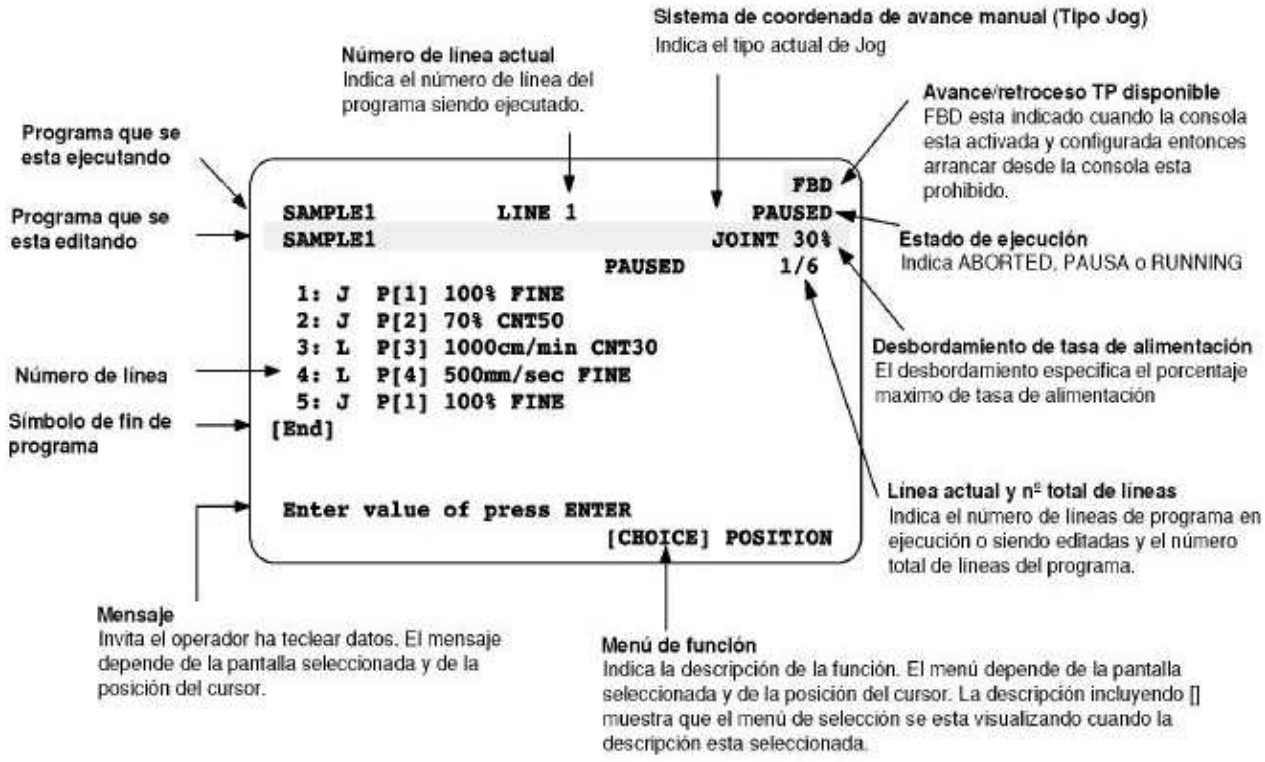
La tecla BACK SPACE borra el carácter o número inmediatamente anterior al cursor.



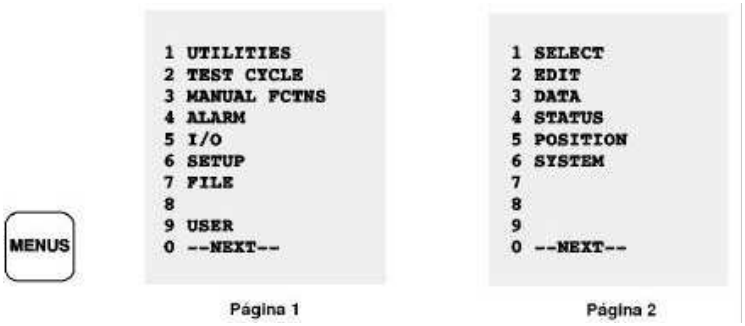
Las teclas del cursor mueven el cursor.

El cursor es la zona destacada que puede moverse en la pantalla de la consola de programación. Esta zona llega a ser el objeto de operación (entrada o cambio del valor o contenidos) de la tecla de la consola de programación.

La tecla ITEM mueve el cursor a una línea cuyo número es especificado.



3.2.1 Fig. 2 Pantalla TPE



3.2.1 Fig. 3 Pantalla TPE: Menus.

En la figura anterior, 3.2.1 Fig. 3, podemos observar las diversas pantallas a las que se accede mediante la tecla MENUS.

UTILITIES La pantalla de utilidad se utiliza para visualizar las pistas.

TEST CYCLE La pantalla de ciclo de prueba se utiliza para especificar los datos para la operación de prueba.

MANUAL FCTNS La pantalla de operación manual se utiliza para ejecutar las instrucciones macro.

ALARM La pantalla del historial de alarma muestra la historia y detalles de las alarmas.

I/O La pantalla de E/S para visualizar, forzar, simular y configurar señales de entrada y salida.

SETUP La pantalla de ajuste se utiliza para establecer el sistema.

FILE La pantalla de archivo se utiliza para leer o almacenar archivos.

USER La pantalla del usuario muestra los mensajes del usuario.

SELECT La pantalla de selección del programa se utiliza para enumerar o crear los programas.

EDIT La pantalla de edición del programa se utiliza para corregir y ejecutar un programa.

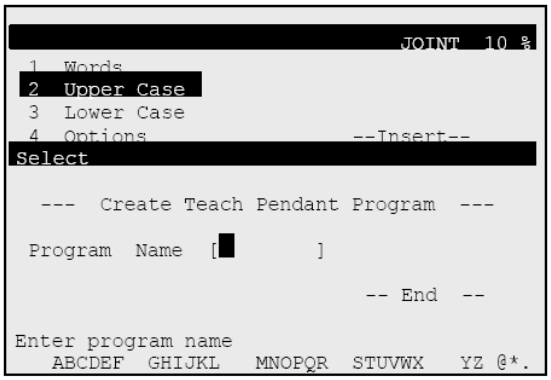
DATA La pantalla de datos muestra los valores en registros, registros de posición,...

STATUS La pantalla de estado muestra el estado del sistema.

POSITION La pantalla de posición actual muestra la posición actual del robot.

SYSTEM La pantalla del sistema se utiliza para establecer las variables del sistema y el mastering.

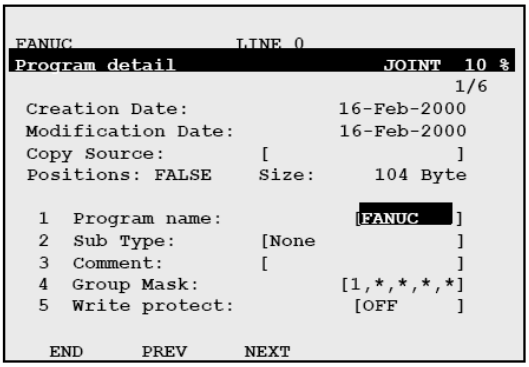
3.2.2 Creación de un programa



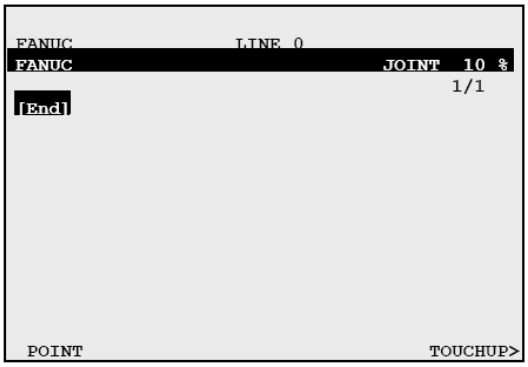
TP en ON, SELECT -> F2: CREATE

Seleccionar el tipo de nombre, validar con ENTER,y después:

->F2 : DETAIL



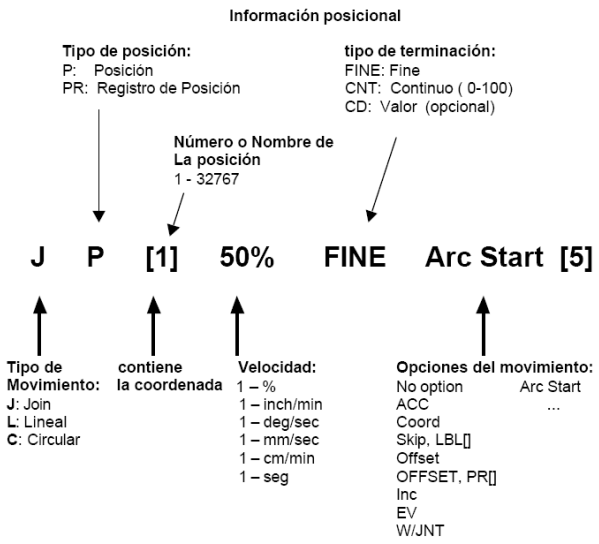
-> F3 : EDIT



3.2.2 Fig. 1,2 y 3 Pantalla TPE: Creación de un programa

3.2.3 Creación de un punto

Desplazar el robot hasta la posición deseada, pulsar SHIFT + F1: POINT



3.2.3 Fig. 1 Descripción de la instrucción de grabación de un punto

3.2.4 Centro de herramienta (TCP)

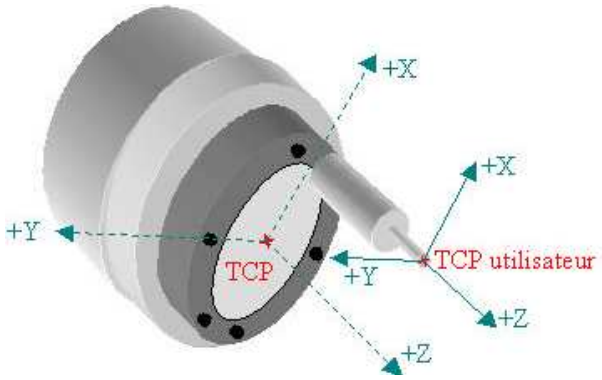
Cada vez que grabamos un punto, éste se nos puede representar en grados y en coordenadas cartesianas. En coordenadas cartesianas, las cotas grabadas, son, de hecho, las del TCP (Tool Center Point = Punto Central de la Herramienta), respecto del origen del sistema de coordenadas cartesianas activo en ese momento y elegido previamente por el usuario. (WORLD por defecto). Por defecto el TCP se encuentra en el centro de la placa del eje 6 del robot.

El TCP es el origen de la referencia herramienta. Cuando se crea una referencia de herramienta, el TCP se desplaza al extremo de la herramienta utilizada. La referencia herramienta puede ser orientada según el eje de ataque de esa herramienta.

Distinguimos 2 tipos de herramientas:

Herramienta simple

Una herramienta simple es una herramienta en la cual el eje de ataque es paralelo al eje Z de la herramienta por defecto. Como observamos en 3.2.4 Fig. 1.

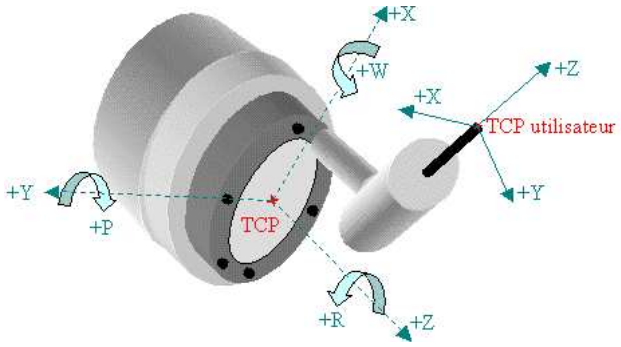


3.2.4 Fig. 1 Herramienta simple

En este caso la orientación de la herramienta no cambia respecto al a herramienta por defecto; solo se desplaza el TCP. El método de aprendizaje de los 3 puntos es el que se elige para memorizar la herramienta.

Herramienta compleja

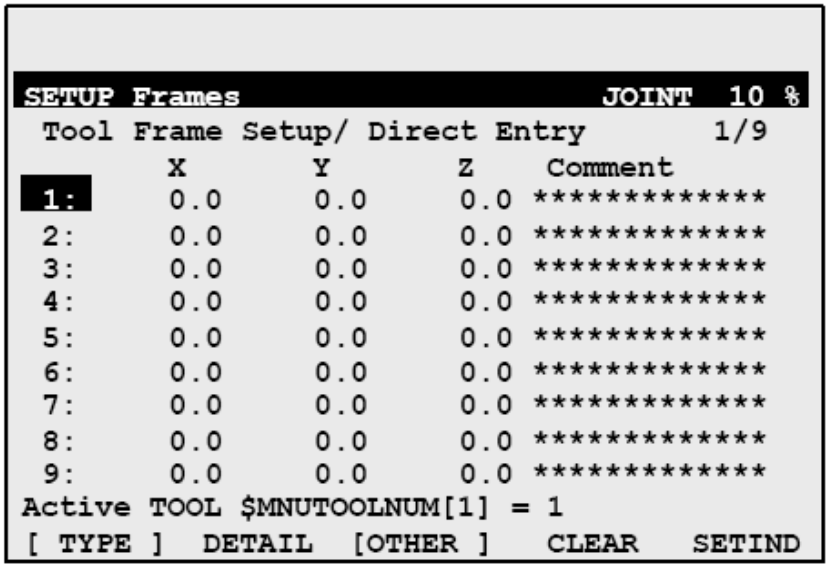
Una herramienta compleja es una herramienta en la cual el eje de ataque no es paralelo al eje Z de la herramienta por defecto.



3.2.4 Fig. 2 Herramienta compleja

En este caso el TCP está desplazado y su orientación está redefinida. Ver 3.2.4 Fig. 2

El método de aprendizaje de los 6 puntos es el que se elige para memorizar la herramienta.



3.2.4 Fig. 3 Pantalla TPE: Frames

Para definir una herramienta seleccionar

MENU -> SETUP -> F1: [TYPE] -> FRAMES -> F3: [OTHER] -> TOOL -> ENTER

Y accederemos a la pantalla represntada en la imagen 3.2.4 Fig. 3.

Es posible definir 10 en R-J3i. Elegir la herramienta a definir con el cursor y después pulsar F2: DETAIL.

Para seleccionar el método de aprendizaje deseado, F2: [METHOD] y después elegir entre los 3 propuestos.

Método de entrada directa de valores

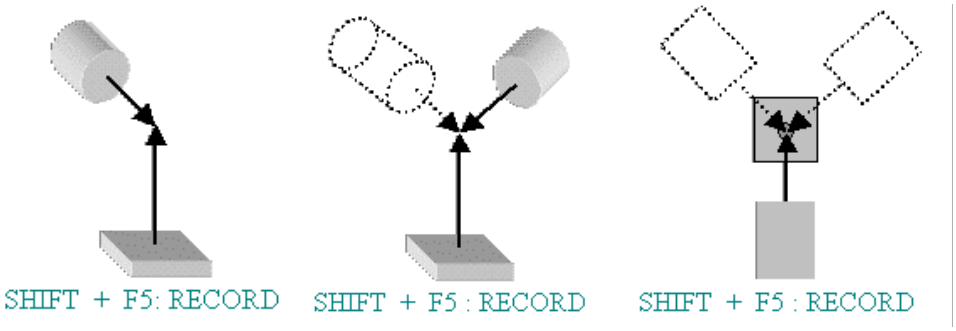
En este método, las coordenadas y orientación de la herramienta a definir deben ser perfectamente conocidos.

Estas coordenadas serán introducidas directamente a mano en la ventana siguiente:

F2: [METHOD] --> DIRECT ENTRY

Método de los 3 puntos

El objeto de este método es el de desplazar el TCP al extremo de la herramienta utilizada. Para ello tenemos que marcar un mismo punto con 3 orientaciones diferentes y memorizar esas posiciones. Proceso representado en 3.2.4 Fig.4.



3.2.4 Fig. 4 Programación de TCP, método de tres puntos.

Cuando los 3 puntos se han memorizado, las coordenadas x, y, z del nuevo TCP, son visualizadas en la parte superior de la ventana. Estas coordenadas son dadas respecto al TCP original de fábrica.

El sentido de la coordenada Z del TCP creado por el método 3P es el mismo que la del TCP original del robot.

Método de los 6 puntos

El objeto de este método es el de desplazar el TCP original del robot a un punto concreto de la herramienta utilizada y de reorientar la herramienta en base a ese punto.

El sentido de la coordenada Z del TCP creado por el método 6P es diferente que la del TCP original del robot. En este caso es impuesta por el usuario.

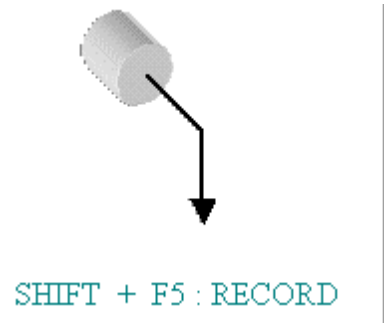
Pasos 1, 2, 3:

Los tres primeros pasos son idénticos a los tres primeros pasos que el método de los tres puntos.

El TCP está definido y ahora debemos re-orientar la herramienta y memorizar tres puntos adicionales.

Paso 4: Orient Origine Point

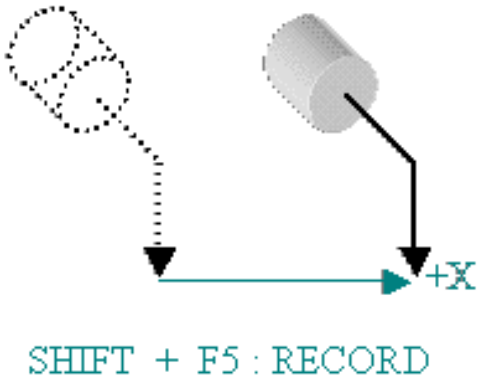
Para memorizar el punto de origen de la orientación, el eje OZ de la herramienta debe estar colocado verticalmente, como en la figura siguiente:



3.2.4 Fig. 5 Programación de TCP, origen.

Paso 5: X Direction Point

Definiremos ahora la orientación y el sentido del eje X. Para este paso y el siguiente, es mas practico moverse en el sistema WORLD, con el fin de asegurar que desplazamos horizontalmente el eje OZ de la herramienta. WORLD → +/- X y/o +/- Y



3.2.4 Fig. 6 Programación de TCP, eje X.

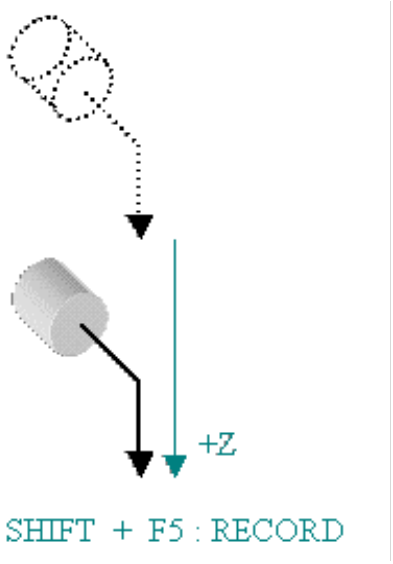
Paso 6: Z Direction Point

Para dar la dirección en Z, es preciso re-posicionarse sobre el punto de origen de la orientación. Para ello colocar el cursor sobre la línea « Orient Origine Point » y después pulsar SHIFT + F4: MOVE_TO.

El robot se re-posicionará sobre el punto memorizado en el paso 4.

Para definir la dirección y el sentido del eje Z.

WORLD → - Z (intentar definir el eje de ataque de la herramienta según el sentido +Z)



3.2.4 Fig. 7 Programación de TCP, eje Z.

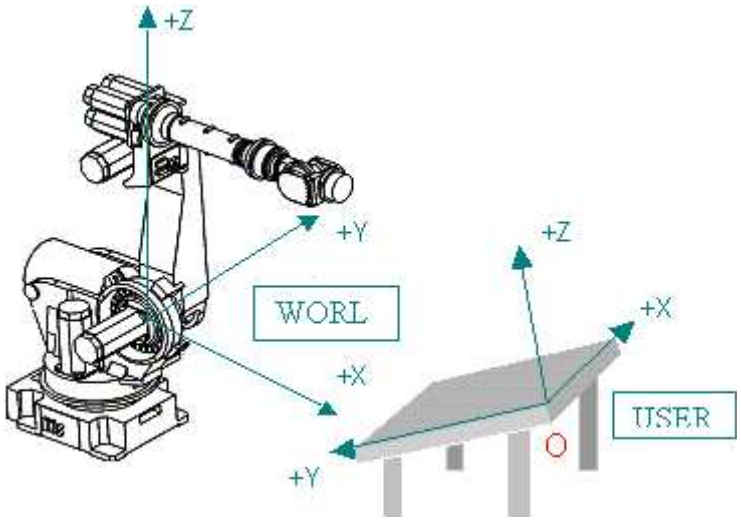
Estado final de la ventana:

Cuando los 6 puntos están memorizados, las coordenadas x, y, z del nuevo TCP y las orientaciones w, p y r de la nueva herramienta son visualizadas en la parte superior de la ventana.

3.2.5 Sistema de referencia usuario

Un sistema de referencia de usuario (UFRAME = USER FRAME) es un sistema de referencia tridimensional, cartesiano sobre el cual se memorizan todas las posiciones de un de terminado programa TP. El TCP se mueve y reorienta en base a ese sistema siempre que movamos el robot en modo USER. Esto se puede ver en la representación gráfica de la figura 3.2.5 Fig. 1.

Si no hay definido ningún sistema de referencia usuario, por defecto, las posiciones se referirán al sistema de coordenadas WORLD.



3.2.4 Fig. 8 Programación de sistema de usuario.

Métodos de configuración

Para definir un sistema de referencia usuario seleccionar:

MENU -> SETUP -> F1: [TYPE] -> FRAMES -> F3: [OTHER] -> USER -> ENTER

La página USER FRAME SETUP aparece.

```
SETUP Frames                          JOINT  10 %
 User Frame Setup/ Direct Entry        1/9
       X         Y         Z     Comment
 1:    0.0       0.0       0.0  ************
 2:    0.0       0.0       0.0  ************
 3:    0.0       0.0       0.0  ************
 4:    0.0       0.0       0.0  ************
 5:    0.0       0.0       0.0  ************
 6:    0.0       0.0       0.0  ************
 7:    0.0       0.0       0.0  ************
 8:    0.0       0.0       0.0  ************
 9:    0.0       0.0       0.0  ************
Active UFRAME $MNUFRAMENUM[1] = 1
[ TYPE ]  DETAIL  [OTHER ]   CLEAR  SETIND >
```

3.2.4 Fig. 9 Pantalla TPE: Configuración Frames.

Elegir la herramienta a definir con el cursor y después pulsar F2: DETAIL.

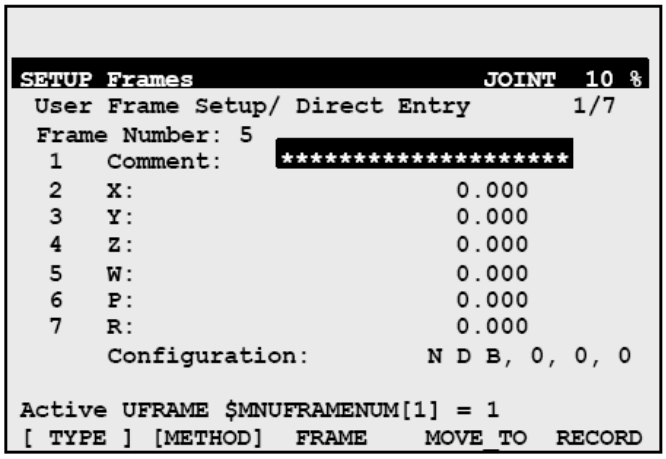
Para seleccionar el método de aprendizaje deseado, F2: [METHOD] y después elegir entre los 3 propuestos.

Para la creación de una referencia de usuario, el origen del sistema de referencia se desplazará al sitio deseado y la posición y orientación siguiendo las 3 direcciones que se elijan.

Método de entrada directa de valores

En este método las coordenadas y orientación de la referencia usuario respecto al WORLD, son perfectamente conocidas. Las coordenadas se introducirán a mano según la ventana siguiente.

F2: [METHOD] → DIRECT ENTRY



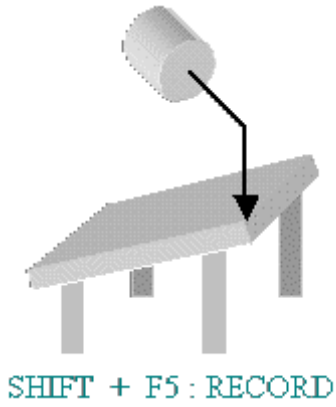
3.2.4 Fig. 10 Pantalla TPE: Programación directa de Frames.

Método de los 3 puntos

F2: [METHOD] → THREE POINT (dos rectas que se cruzan determinan un plano, con origen fijo en el punto de cruce y Z perpendicular al plano.)

Paso 1: Orient Origine Point

Para el primer paso memorizaremos el origen de la referencia.




```
SETUP Frames                          JOINT  10 %
 User Frame Setup/ Three Point          2/4
 Frame Number: 1

   X:    0.0    Y:    0.0    Z:    0.0
   W:    0.0    P:    0.0    R:    0.0

 Comment:********************
 Orient Origin Point:   UNINIT
 X Direction Point:     UNINIT
 Y Direction Point:     UNINIT

Active UFRAME $MNUFRAMENUM[1] = 1
[ TYPE ] [METHOD]  FRAME    MOVE_TO  RECORD
```

3.2.4 Fig. 11 y 12 Programación Sistema de usuario, Origen.

3.2.6 Instrucciones con registros y registros de posición

Las variables disponibles a utilizar son:

Los registros: real (32 bits) o entero

Los registros de posición: puntos en coordenadas joint, puntos en coordenadas cartesianas o matrices.

Estas son variables globales (todos los programas tienen acceso a todos los registros y registros de posición).

Los registros

Hay un máximo de 256 (configurables). Un registro permite ser comentado con un nombre.

El direccionamiento puede ser:

Directo

R[1] = 2 -> el valor es guardado directamente en R[1]

O indirecto

R[R[1]] = 5 -> el registro afectado depende del valor contenido en R[1]

Si R[1] = n, por tanto el valor 5 es guardado en R[n].

· En un registro es posible almacenar el resultado de una operación aritmética.

R[n] = [valor] [operador] [valor]

- El [operador] puede ser: - una suma (+)
- una resta (-)
- una multiplicación (*)
- una división (/)
- una división entera (DIV)
- el resto de una división (MOD)
- El [valor] puede ser: - una constante
- un valor de entrada-salida analógico AI[n]/AO[n]
- un valor de entrada-salida digital DI[n]/DO[n]
- un valor de entrada-salida grupo GI[n]/GO[n]
- un valor de entrada-salida de robot RI[n]/RO[n]
- un valor de un registro R[n]
- un valor de un elemento de un registro de posición PR[i , j]

Para insertar en un programa -> F1: [INST] -> Registers.

Para visualizar la lista de registros y su contenido -> DATA -> F1: [TYPE] -> Registers.

Los registros de posición

Un registro de posición almacena un punto.

El direccionamiento puede ser:

Directo

PR[1] = P[1] -> el punto es guardado directamente en PR[1]

O indirecto

PR[R[1]] = P[3] -> el registro de posición afectado depende del valor contenido en R[1]

Si R[1] = n, entonces el punto P[3] está almacenado en PR[n].

En un registro de posición es posible almacenar un punto o una operación de punto.

PR[n] = [punto] [operador] [punto]

- El [operador] puede ser:

- una suma (+)
- una resta (-)
- El [punto] puede ser:
- una posición P[n]
- un registro de posición PR[n]
- la posición actual del robot en grados eje por eje JPOS
- la posición actual del robot en cartesianas LPOS

Los registros de posición son también accesibles elemento por elemento.

Por ejemplo, la coordenada j de PR[i] está definida por PR[i , j]

PR[1,2] = 300 -> la coordenada Y de PR[1] está inicializada a 300mm.

O indirectamente

R[1] = 1

R[2] = 2

PR[R[1],R[2]] = 300 -> la coordenada Y de PR[1] está inicializada a 300mm.

Cada posición y orientación es por tanto accesible independientemente.

Para insertar en un programa ->1: [INST] ->registers

Para visualizar la lista de registros y su contenido -> DATA -> F1: [TYPE] -> Position Registers.

3.2.7 Instrucciones de salto condicional

Una instrucción de salto condicional permite efectuar un salto (o bucle) a una etiqueta situada en el mismo programa si (y sólo si) ciertas condiciones son verdaderas.

F1: [INST] ->F/SELECT.

Instrucción IF-

Efectúa un salto en función de una condición verdadera IF [valor1] [operador] [valor2] [salto]

El [valor1] puede ser:

- un valor de un registro R[n]
- un valor de entradas-salidas analógicas AI[n]/AO[n]
- un valor de entradas-salidas digitales DI[n]/DO[n]
- un valor de entradas-salidas de grupo GI[n]/GO[n]
- un valor de entradas-salidas de robot RI[n]/RO[n]

El [operador] puede ser:

- un test de igual (=)
- un test de diferente (<>)
- un test de menor (<)
- un test de mayor (>)
- un test de menor o igual (<=)
- un test de mayor o igual (=>)

El [valor2] puede ser:

- una constante
- ON
- OFF
- un valor de un registro R[n]
- un valor de entradas-salidas analógicas AI[n]/AO[n]
- un valor de entradas-salidas digitales DI[n]/DO[n]
- un valor de entradas-salidas de grupo GI[n]/GO[n]
- un valor de entradas-salidas de robot RI[n]/RO[n]

El [salto] puede ser:

- un JMP LBL[n]
- un CALL programa

Instrucción SELECT-Efectúa uno o varios saltos en función del valor de un registro.
SELECT R[n] = [valor 1], [salto]
[valor 2], [salto]
[valor n], [salto]
ELSE, [salto]

Los [valores] pueden ser:

- una constante
- un valor de un registro R[n]

Los [saltos] pueden ser:

- un JMP LBL[n]
- un CALL programa

No olvidar ELSE como fin de instrucción, ya que tiene en cuenta todos los valores posibles del registro
R[n] no citados.

3.2.8 Instrucciones de espera

Las instrucciones de espera retardan la ejecución de un programa mediante un tiempo especificado o hasta que una condición sea verdadera.

F1: [INST] -> WAIT.

Temporización- Retarda la ejecución de un programa durante un tiempo especificado. La duración se expresa en segundos; hay un mínimo de 0,01 segundos WAIT [tiempo]. El [tiempo] puede ser:
- una constante
- un registro R[n]

Espera de una condición verdadera- Retarda la ejecución de un programa hasta que la condición sea verdadera. WAIT [valor 1] [operador] [valor 2] [tiempo].

- El [valor] puede ser:

- un valor de un registro R[n]
- un valor de entradas-salidas digitales DI[n]/DO[n]
- un valor de entradas-salidas de robot RI[n]/RO[n]

- El [operador] puede ser:

- un test de igual (=)
- un test de diferente (<>)
- El [valor 2] puede ser:
- una constante
- ON
- OFF
- un valor de un registro R[n]
- un valor de entradas-salidas digitales DI[n]/DO[n]
- un valor de entradas-salidas de robot RI[n]/RO[n]

- El [tiempo] puede ser:

- FOREVER -> espera mientras la condición no se cumpla
- TIMEOUT LBL[n] -> espera el tiempo especificado en la variable timeout ($WAITTMOUT), después salta a label n si la condición no se ha cumplido.

3.2.9 Registros periféricos de I/O (UOP)

Registros de entrada (UI)-

***IMSTP** Siempre activa	*IMSTP es la señal de parada inmediata del software.*IMSTP es una señal normal de OFF que se mantiene en ON. Cuando se pone a OFF sucede que: - Se detiene el programa si se estaba ejecutando. - Se detiene inmediatamente el robot y actúan los frenos. - Se desconecta la tensión de los servos Cuando se pierde esta señal, se visualiza el código de error SRVO-037 *IMSTP input. Esta señal esta siempre activa
***HOLD** Siempre activa	*HOLD es la señal de detención externa. *HOLD es una señal de normal de OFF, que se mantiene en ON. Cuando se pone en OFF, sucede lo siguiente: - Detiene le programa en ejecución. - Desciende la velocidad a una controlada y se detiene. - El freno en detención desconecta la tensión de los servos al parar el robot.
***SFSPD** Siempre activa	*SFSPD es la señal de entrada de velocidad de seguridad. Esta señal se conecta generalmente a la valla de seguridad.*SFSPD es una señal normal de OFF, que se mantiene en ON. Cuando pasa a OFF, sucede lo siguiente: - Detiene el programa en ejecución. - Reduce la velocidad a un valor definido en una variable del sistema. Este valor no puede incrementarse mientras *SFSPD este en OFF. - Visualiza el mensaje de error SYST009 - No permite una condición de marcha REMOTE. Las entradas de marcha del UOP o del SOP se inhabilitan cuando SFSPD se pone en OFF y tan solo el terminal de enseñanza tiene control sobre el movimiento con la velocidad limitada.
FAULT_RESET Siempre activa	FAULT_RESET es la señal externa de aceptar el fallo. Cuando se recibe esta señal, sucede lo siguiente: - Cesa el estado de error. - Los servos reciben tensión. - El programa detenido no continuará.
START Activa cuando el robot se halla en condición de mando remoto (CMDENBL=ON)	START es la entrada de marcha remita. La función de esta señal, depende de la variable del sistema $SHELL_CFG.$CON_ONLY se fija a FALSE, la señal de START: - Hace seguir a un programa detenido. - Si el programa se aborta, el programa seleccionado en este momento empieza en la posición del cursor.

Registros de Salida (UO)

CMDENBL	CMDENBL es la salida de habilitación de órdenes. Esta salida indica que el robot se halla en una situación de mando remoto. Esta señal se pone en ON cuando en interruptor remoto se conmuta a ON. Esta salida solamente permanece activa cuando el robot no se halla en situación de fallo. Cuando SYSRO esta en OFF, CMDENBL está en OFF. Esta señal se pone activa cuando se cumplen todas las condiciones siguientes: - Terminal de enseñanza inhabilitado - Interruptor remoto en ON - Entrada SFPD en ON - Entrada ENBL en ON - Variable del sistema $RMT_MASTER a 0 - No se halla en modo paso a paso (STEP) - Interruptor de selección en modo Auto.
SYSRDY	SYSRDY es la salida de sistema preparado. Esta salida indica que los servos se hallan activos.
PROGRUN	PROGRUN es la salida de programa funcionando. Esta salida se pone activa cuando hay un programa funcionando.
PAUSED	PAUSED es la salida de programa detenido. Esta salida se pone activa cuando se ha detenido un programa.
HELD	HELD es la salida de detención. Esta salida se pone activa cuando la entrada *HOLD del UOP se halla en OFF o se ha accionado el pulsador HOLD de la TP
FAULT	FAULT es la salida de error. La salida se pone activa cuando un programa entra en situación de error.

Estas señales permiten comandar el robot a distancia por medio de un panel de operador (UOP) o PLC. Las funciones de las salidas UOP (UI[n] UO[n]) están predefinidas y pueden ser cableadas sobre cartas modulares digitales o configuradas mediante cartas de bus de campo (Interbus, Profibus, Devicenet,...) 18 entradas y 20/24 salidas (4 opcionales) pueden ser conectadas (mínimo 8 entradas o salidas).
Para llegar a la pantalla de configuración de las entradas y salidas del robot deberemos pulsar:
<Menú> → 5 I/O pulsando F1[TYPE] vemos los diferentes tipos de entradas y salidas.
1
2 Digitales
3
4 Grupos
5
6 UOP

En cualquier pantalla de entradas y salidas, si pulsamos F2 [CONFIG] entramos dentro de la pantalla de configuración de las mismas.

UOP:

Los parámetros de estas I/O son iguales que los de las I/O digitales; hay que tener en cuenta que los rangos de estos dos grupos no se pueden superponer en el mapa de I/O;

además, también hay que recordar que las UOP marcadas con un asterisco (*) funcionan con lógica negada.

Usando la tecla F3 [IN/OUT] alternamos la pantalla de configuración de entradas con la de salidas y viceversa.

3.2 Configuración del Robot

3.3.1 Pantalla de configuración del sistema

MENUS, 0- NEXT, 6- SYSTEM, F1- TYPE, Config.

Accedemos a la pantalla de configuración del sistema representada en 3.3.1 Fig. 1.

```
System/Config                          JOINT 30%
                                              1/27
 1 Use HOT START:                          FALSE
 2 I/O power fail recovery: RECOVER ALL
 3 Autoexec program                 [********]
         for Cold start:
 4 Autoexec program                 [********]
         for Hot start:
 5 HOT START done signal:                  DO[0]
 6 Restore selected program:               TRUE
 7 Enable UI signals :                     TRUE
 8 START for CONTINUE only :              FALSE
 9 CSTOPI for ABORT :                     FALSE
10 Abort all programs by CSTOPI :  FALSE
11 PROD_START depend on PNSTROBE :FALSE
12 Detect FAULT_RESET signal :       FALL
13 Use PPABN signal :              <*GROUPS*>
14 WAIT timeout :                   30.00 sec
15 RECEIVE timeout :                30.000 sec
16 Return to top of program :             TRUE
17 Original program name (F1) : [PRG     ]
18 Original program name (F2) : [MAIN    ]
19 Original program name (F3) : [SUB     ]
20 Original program name (F4) : [TEST    ]
21 Original program name (F5) : [*******]
22 Default logical command :   <*DETAIL*>
23 Muximum of ACC instruction :    150
24 Minimum of ACC instruction :      0
25 WJNT for default motion :       ******
26 Auto display of alarm menu :    FALSE
27 Force Message :                ENABLE
28 Reset CHAIN FAILURE detection : FALSE
29 Allow Force I/O in AUTO mode  : TRUE
30 Allow chg. ovrd. in AUTO mode : TRUE
31 Signal to set in AUTO mode DOUT [  0]
32 Signal to set in T1 mode DOUT   [  0]
33 Signal to set in T2 mode DOUT   [  0]
34 Signal to set if E-STOP DOUT    [  0]
35 Hand broken :                 <*GROUPS*>
36 Remote / Local setup :          Remote
37 External I/O (ON : Remote) : DI [  0]

[TYPE]                    [CHOICE]
```

3.3.1 Fig 1 Pantalla TPE: Configuración del sistema

3.3.2 Configuración de la red profibús

Accedemos a la pantalla de configuración de la red profibús mediante las teclas:
<Menu>→ 6 Setup
F1 [TYPE] → 0 NEXT → 7 PROFIBÚS
F3 [OTHER]

1– Slave- Configuración del profibús del esclavo
2- Master- Configuración del profibús del master.
3- Bus Param.- Configuración de los parámetros del profibús.
4- Slave Param.- Pantalla de configuración de los parámetros de los diferentes esclavos del robot.

Dentro de la pantalla Slave Param. Pulsando F2 [DETAIL] habilitamos/deshabilitamos el esclavo, configuramos su dirección, establecemos el nombre, bytes de entrada y salida, flags…

3.3.3 Configuración de las entradas y salidas

I/O digitales:
Rack- Dirección de la tarjeta de I/O del robot (66→ Master; 67 → Slave
Slot- Lugar físico donde se encuentra situada la tarjeta de I/O.
Range- Número de I/O "asociadas".
Start- Número de bit por el que empieza el rango de I/O.

Grupos:
Rack- Dirección de la tarjeta de I/O del robot (66→ Master; 67 → Slave
Slot- Lugar físico donde se encuentra situada la tarjeta de I/O.
Range- Número de I/O "asociadas".
Start- Número de bit por el que empieza el rango de I/O.

3.3.4 Configuración del Payload: la Garra

Para configurar un Payload tendremos que acceder a la pantalla de sistema:
<Menú> → 0 NEXT → 6 SYSTEM
F1 [TYPE] → 6 MOTION
Configuramos el peso, la posición del centro de gravedad y le asignamos un nombre. Tenemos hasta un máximo de 10 payloads diferentes.

3.3.5 Configuración de Macros

Una MACRO es un programa que efectúa una operación específica cuya ejecución puede ser comandada por:

· la activación de una tecla de usuario del Teach Pendant (UK[n]). Group Mask (*;*;*;*;*)

· la activación de una tecla de usuario del Teach Pendant SHIFT + (SU[n])

· la activación de una tecla de usuario del controlador (opción) (SP[n])

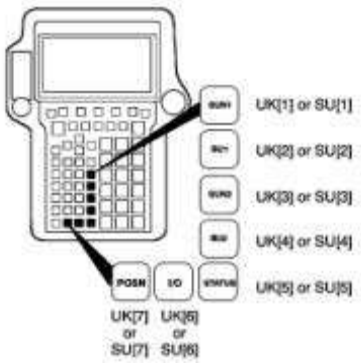
· la selección de un ítem del menú MANUAL FCTNS (MF[n])

· instrucción CALL

· instrucción RUN

· la activación de una entrada (DI[n]/RI[n]). Para ampliar $MACROMAXDRI.

· la activación de una entrada UI[n].



3.3.5 Fig 1 Teclas de Macro del TPE

Si tenemos programas guardados en formato macro, existen varias teclas que pueden ser configuradas de manera que al ser pulsadas ejecuten el programa asignado.

<Menú> → 6 SETUP

F1 [TYPE] –4> 4 Macro

Instruction name: Comentario sobre la macro a ejecutar

Program: Con la tecla F4[CHOICE] seleccionamos la macro a ejecutar.

Assign: Aquí escogemos la tecla a usar y la forma de llamar a la macro requerida.

F4 [CHOICE]

UK- User Key.

SU- Shift User Key.

Mf- Manual Funtion.

3.3.6 Ajuste de límite de ejes

Existen 3 tipos de limitación del recorrido de ejes:

- Límites de software.

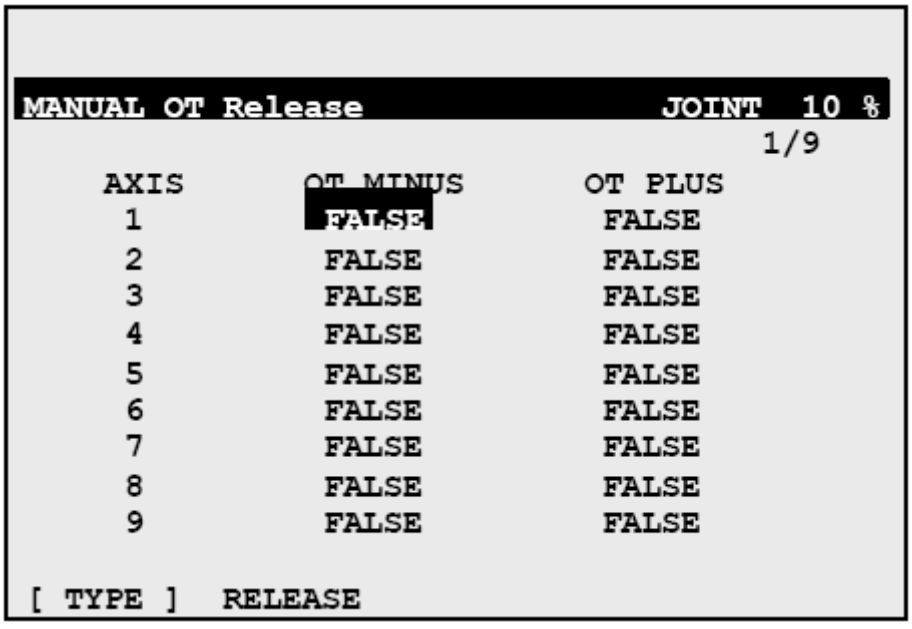
- Límites eléctricos.

- Límites mecánicos.

Los límites de software fijos

Estos son los primeros límites que se encuentra el robot (si están correctamente definidos). Cuando un límite de software es alcanzado, el robot no da fallo, simplemente se para y no permite movimiento en ese sentido. Para poder volver a mover el robot, es suficiente con mover el robot en sentido inverso. Si queremos acceder a la pantalla de la imagen 3.3.6 Fig. 1 deberemos proceder como se indica a continuación:

MENU, 0-NEXT, 6-SYSTEM, F1-[TYPE], Axis limits.



3.3.6 Fig 1 Pantalla TPE: limite de ejes

Para que las modificaciones de límite de software se tengan en cuenta, es preciso apagar y volver a arrancar el controlador.

Límites mecánicos

Es posible reglar ciertos límites mecánicos; eso depende de los ejes y de los robots. Si sucede que un límite mecánico es alcanzado, se deben verificar los límites eléctricos y los límites de software.

Normalmente dará una alarma de colisión por sobre consumo de motor.

3.3.7 Arranque de programa a distancia vía UI

[6:START]

Para utilizar las UOP se debe respetar el siguiente protocolo:

Configurar las señales del sistema UOP. (ver capítulo de configuración)

Cablear las señales del sistema obligatorias y las que se deseen para control de la instalación.

Para que la señal de entrada UI [6:START] tenga efecto se han de cumplir dos condiciones:

1ª- Habilitar las UI signals:

MENU, O-NEXT, 6-SYSTEM, F1-TYPE, 5-CONFIG, ENABLE UI SIGNALS a "TRUE".

2ª- El robot nos tiene que dar la señal de salida UO [1:CMD ENABLE]=ON:

¿Cuándo el robot pone la salida UO [1:CMD ENABLE] a ON?

1-UI [1:*IMSTP]=ON, no se recibe ninguna emergencia externa por software.

2-UI [2:*HOLD]=ON, no se recibe ningún paro de programa externo.

3-UI [3:*SFSPD]=ON, no se recibe ningún paro de programa asociado a un arranque con velocidad predefinida en una variable.

4-UI [8:*ENABLE]=ON, se permite la habilitación de movimientos al robot.

5-Llave T1, T2, AUTO se encuentra en modo AUTO, con lo que las seguridades externas por hardware quedan habilitadas.

6-Controlador en modo REMOTO con lo que se permite el arranque del robot desde un sistema remoto por ejemplo un pulsador de "marcha" asociado a la entrada UI [6:START] que generará un pulso que tendrá su efecto con el flanco descendente.

Para ello:

R-J2 y R-J3 à Llave LOCAL / REMOTE en REMOTE

R-J3i à Menú, 0-Next, 6-System, F1-Type, Config, línea 36, Opción

Local/Remote = Remote

7-Variable del sistema $RMT_MASTER=0 si no lo está.

MENU, O- NEXT, 6- SYSTEM, F1- TYPE, 2- VARIABLES.

8-Asegurarse opción *Start For Continue Only* está a false:

MENU, O- NEXT, 6- SYSTEM, F1- TYPE, 5- CONFIG, START FOR CONTINUE ONLY a "False".

(si estaba a "True", poner a False y luego OFF/ON para que tome efecto, o modificar la variable $SHELL_CFG.$CONT_ONLY=FALSE)

9-Teach Pendant en OFF y en condiciones de no STEP (paso a paso).

10-UO [2:SYS READY]=ON, el robot no tiene ningún fallo.

Reset de fallos: Reset de fallos externos vía software a traves de las UI's.

Reset de fallos externos vía hardware (Emerg. Externas, Fence Open,...)

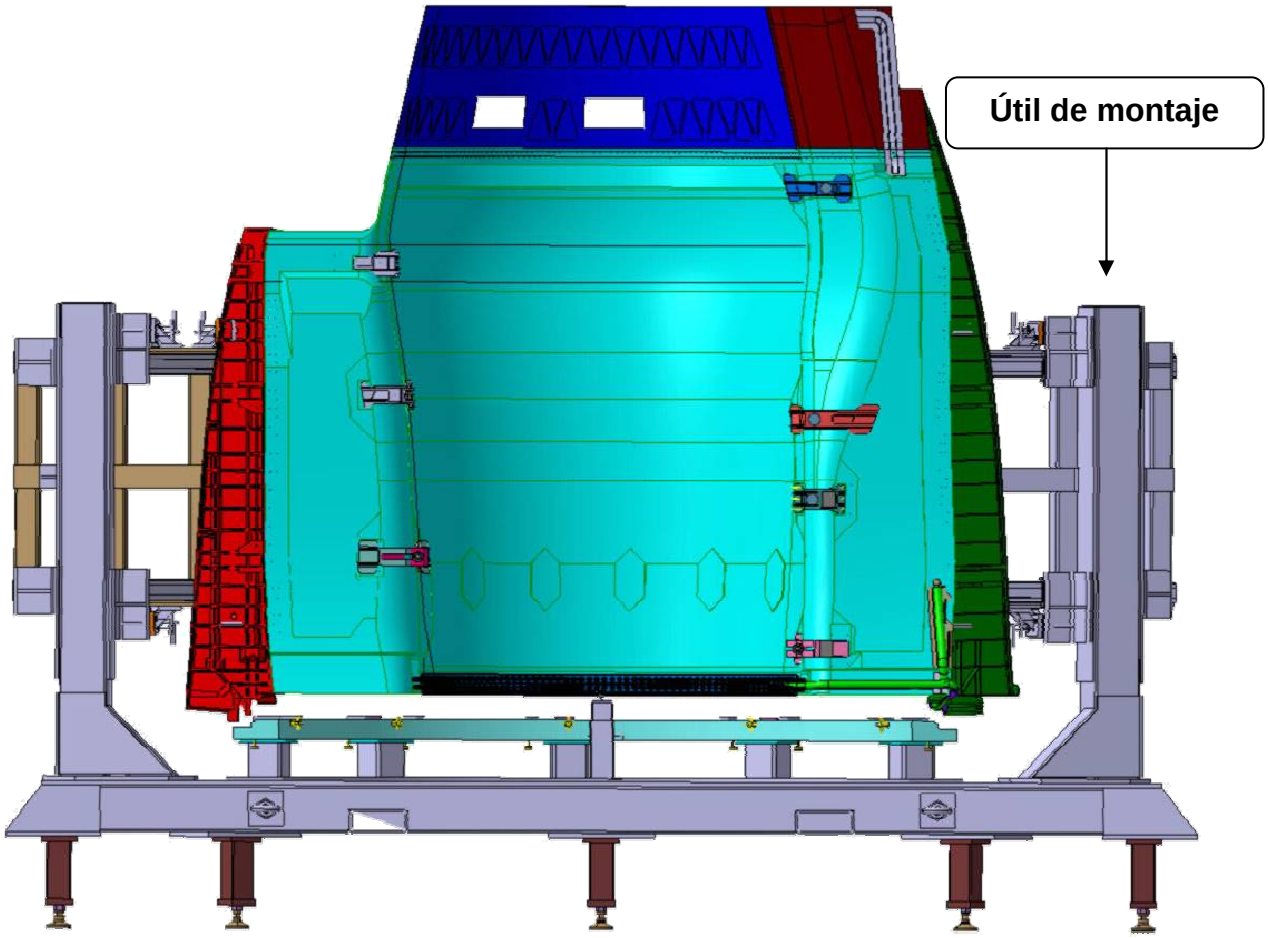
11-FCTN, 1-Abort All, Select, seleccionar el programa arrancar.

12- UI[6:Start] tiene su efecto sobre el robot con flanco descendente.

4 PROGRAMACIÓN DE LA APLICACIÓN

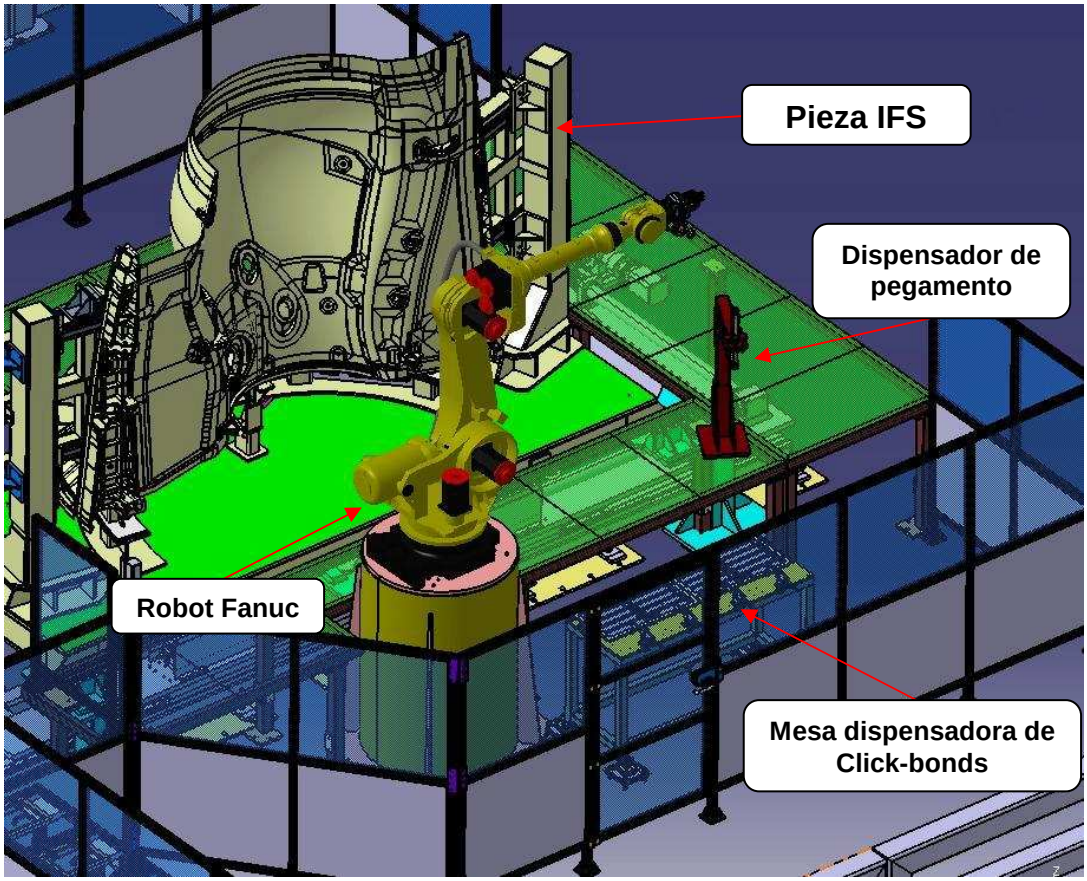
Para poder realizar la programación de la aplicación es primordial saber cómo se realizará el proceso de posicionamiento de los click-bonds sobre la pieza; desde el momento en que tenemos lista la pieza hasta que damos el proceso por finalizado.

La célula de montaje automatizada, instalará fijaciones (click-bonds) en distintos componentes en el interior del montaje de la estructura fija mediante un robot Fanuc R3000*i*.



4 Fig. 1. Carenado del motor y útil

Estas sujeciones servirán para fijar una tela asfáltica en la parte interior de la pieza representada en 4 Fig1, con el objetivo de aislar térmicamente este componente de las elevadas temperaturas que llega a alcanzar el reactor en su interior.



4 Fig. 2. Célula posicionadora de Click-bonds

La Celula para el posicionamiento de los fastener consta de:

- Una mesa para la dispensación de los Click-bonds, y herramientas de lijado, limpieza y secado; de carga manual.
- Un motor para la regulación del dispensador de adhesivo.
- Dispensador de Acetona.
- Robot Fanuc 3000*i*.
- Vallado de seguridad controlado por PLC.

Cada uno de estos elementos podemos verlos representados en la imagen anterior (4. Fig. 2)

4.2 Proceso

Antes de iniciar el proceso del robot, deben cumplirse varias condiciones iniciales controladas por el PLC:

- Debe haber pieza colocada en la zona del robot.
- La mesa del robot debe estar correcta; herramientas y click-bonds suficientes para realizar la operación.
- La zona del robot debe encontrarse ok; puertas cerradas y barreras rearmadas.

Una vez estas condiciones se cumplan, el operario debe dar la señal de Start al robot para que comience a correr el programa inicial del robot (main).

El robot debe preparar cada punto de inserción de click-bond lijándolo y limpiándolo con las herramientas establecidas para ello.

Después del proceso de lijado, procederá a la limpieza y secado de tantos puntos como sea necesario, desde un mínimo de 1 hasta un máximo igual al total de puntos de la pieza

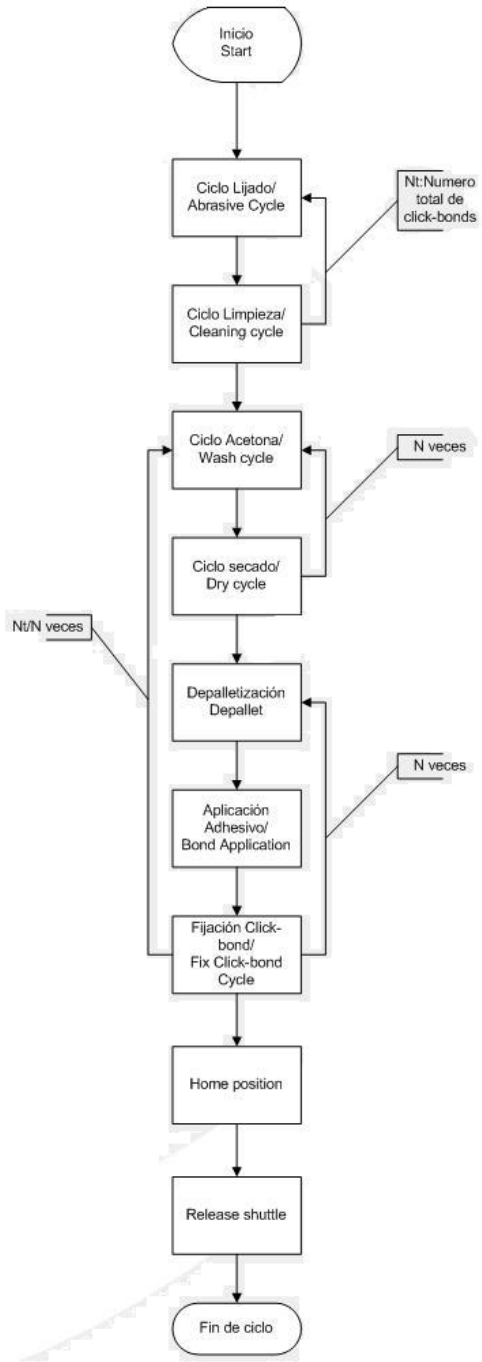
Una vez realizada esta operación, el robot esperará a que el operario realice la carga de pegamento de forma manual, rearme la zona y vuelva a darle la señal de Start.

Una vez recibido el Start, el robot continuará el proceso, aplicando pegamento al click-bond y posicionándolo en el punto adecuado.

La situación de los puntos en los que posicionar los click-bonds, se encuentran almacenados en un programa aparte, para que, en el caso de alguna modificación de la posición de uno o varios puntos, solo deba ser modificada dicha base de datos de puntos; evitando que una modificación pueda afectar al resto del proceso de una manera significativa.

Cuando el robot finaliza el proceso, da una señal de final de trabajo que permite la retirada de la pieza y el ingreso de una nueva.

Diagrama general



4.1 Fig. 1 Diagrama general del proceso

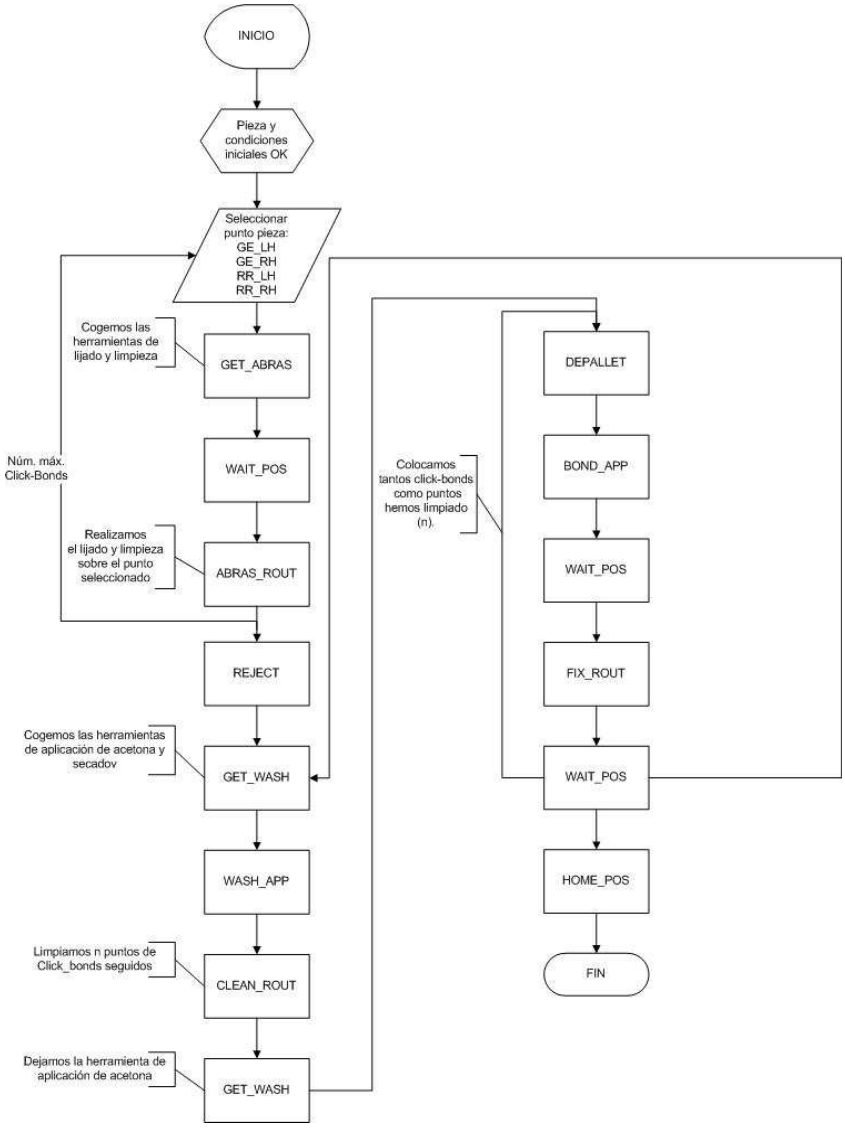
Este diagrama de flujo, nos muestra los diferentes trabajos que realiza el robot; se puede diferenciar tres ciclos: Lijado, limpieza y fijado.

En el ciclo de lijado el robot procede a coger las herramientas necesarias para lijar y limpiar uno por uno todos los puntos y después desechar las herramientas usadas.

En el ciclo de limpieza, el robot procede a recoger las herramientas de limpieza y aplicar acetona y secar la acetona de tantos puntos como hayamos configurado. Una vez se ha limpiado el número de puntos configurado, se procede a dejar en el parking la herramienta que aplica la acetona.

En el ciclo de fijado, se procede a depaletizar el elemento a fijar, aplicarle la cola especial para fibra de carbono, (la aplicación de la cola está controlada por el PLC), y fijarlo en la pieza. Este proceso se repite tantas veces como puntos han sido limpiados en el ciclo de limpieza.

Los ciclos de limpieza y fijado, se realizan tantas veces sea necesario hasta haber limpiado y fijado el número total de puntos configurados.



4.1 Fig 2 Diagrama de flujo de la aplicación

En el diagrama de flujo anterior (4.1 Fig.2). se pueden ver los tres ciclos comentados anteriormente (lijado, limpieza y fijado) y los programas que participan en cada uno de ellos.

Podríamos llegar a diferenciar tres tipos diferentes de programas:

Programa de movimiento; es un programa realizado mediante el método de enseñanza, situando el robot en una posición y memorizándola.

Programa de puntos; es un programa realizado en off-line, donde extraemos los puntos de un fichero grafico.

Programa de posición; es un programa realizado para acceder a los distintos puntos del programa de puntos usando un registro de posición.

4.1.1 Programas de movimiento

A este tipo de programa pertenecerían Home_Pos, Wait_Pos, Bond_App y Wash_App. Estos, son programas realizados memorizando la posición deseada mediante el Teach pendant; siguen la estructura siguiente:

1: !Go to wait position ;	
2: ;	
3: UTOOL_NUM = 1 ;	Declaramos la herramienta con la que trabajaremos.
4: PAYLOAD[1] ;	Declaramos la carga.
5: UFRAME_NUM = 2 ;	Declaramos el marco en el que estará referenciado el punto memorizado.
6:L P[1:Wait position] 1500mm/sec CNT50 ;	
7:J P[2:Wait position] 100% CNT50 ;	

Los puntos P1 y P2 son los puntos guardados mediante este método; hay que tener en cuenta que tanto la herramienta (UTOOL) y el marco (USERFRAME) del programa y de los puntos deben coincidir. Con esto conseguimos evitar los posibles fallos de programa debido a los límites del eje 4 y 6.

4.1.2 Programas de puntos

En este tipo de programa es donde se encuentran guardados los puntos donde se posicionara el elemento y donde se carga el punto en el registro de posición con el que vamos a trabajar.

La estructura a seguir para realizar los programas de puntos es la siguiente:

1: !GE left hand. Ref.721Z3700 ;	
2: ;	En este registro guardamos el número total de Clickbonds a colocar.
3: !Clickbonds ;	
4: R[15] = 200;	
5: !Clickbonds in work zone 1 ;	En este registro, se guardan el numero de puntos pertenecientes a la zona de trabajo 1
6: R[16] = 50;	
7: !Clickbonds in work zone 2 ;	En este registro, se guardan el numero de puntos pertenecientes a la zona de trabajo 2
8: R[17] = 100 ;	
9: ;	
10: PR[3] = P[R[11]] ;	En esta función, podemos ver que guardamos en el registro de posición 3 la posición perteneciente al punto donde señale el registro 11, utilizándolo a modo de puntero
11: END ;	
12: ;	
13: !**********;	
14: ! ;	
15: ! ATENTION !!! ;	
16: ! ;	
17: ! Only click bond positions ;	
18: ! below this line ;	
19: ! ;	

20: ! NEVER PLAY IN AUTO MODE ;	
21: ! ;	
22: !**********;	
23: END ;	
24: ;	
25: !Click bond pos for Ref.721Z3700 ;	
26: ;	
27: UTOOL_NUM = 1 ;	
28: UFRAME_NUM = 1 ;	
29: ;	A partir de esta línea, se encuentran los puntos donde deberán ser colocados los clickbonds; tantos como nos diga el registro R(15)
30:L P[1] 500mm/sec FINE ;	
31: END ;	

Al ser cada punto usado de forma reiterada en diversos programas de la aplicación, se consigue simplificar la modificación de la posición de uno o varios puntos; ya que realizando la modificación en el programa de puntos pertinente, quedara reflejado en la ejecución de los diversos programas que usen dicho punto (programas de posición).

Contamos con cuatro programas diferentes que pertenecen a este tipo; cada programa tiene almacenados los puntos pertenecientes a una de los cuatro modelos diferentes que existen. Estos programas son: RR_LH, RR_RH, GE_LH y GE_RH.

4.1.3 Programas de posición

En este tipo de programa, accedemos a las posiciones predefinidas mediante un registro de posición y la modificación de los TCP's.

Aquí englobaríamos los siguientes programas:

Get_abras, Get_wash, Nget_wash, Fix_rout, Clean_rout, Reject.

La estructura para el acceso a los diferentes puntos es la siguiente:

10: LBL[10:Start]	
11: !Dry section ;	
12: PR[11] = UTOOL[1] ;	Cargamos en los PR 11 y 13 el TCP de la herramienta a modificar.
13: PR[13] = UTOOL[1] ;	
14: ;	
15: !Distance calculation ;	
16: PR[GP1:13,10] = PR[13,10] - 0 ;	Modificamos en X la posición de la herramienta
17: PR[GP1:13,11] = PR[13,11] + 0 ;	Modificamos en Y la posición de la herramienta
18: PR[GP1:13,12] = PR[13,12] + 140 ;	Modificamos en Z la posición de la herramienta

En este punto, el TCP es 140 milímetros más largo.

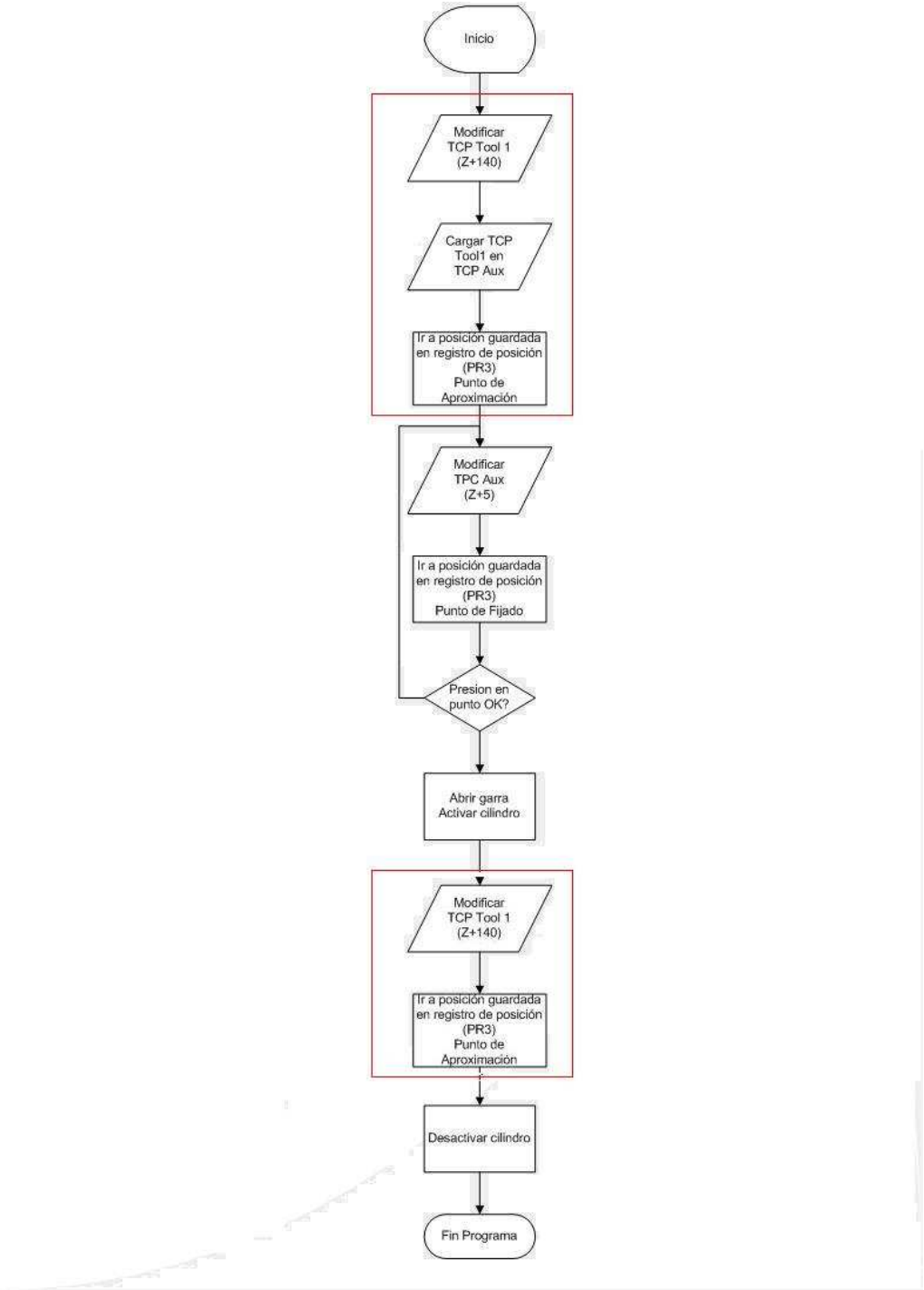
19: !Aproching point ;	
20: UTOOL[4] = PR[13] ;	Cargamos el TCP modificado en el Tool 4; tool auxiliar.
21: UTOOL_NUM = 4 ;	Activamos el Tool 4
22:L PR[3] 1000mm/sec FINE ;	Mandamos al robot al punto guardado en el registro de posición

Al ser el TCP activado 140 milímetros más largo que el TCP con el que se grabó el punto guardado en el registro de posición, la posición real de la herramienta es 140 milimetros separado en Z del punto deseado.

24: !Fixing point ;	
25: UTOOL[4] = PR[11] ;	Cargamos el TCP real en el Tool 4; tool auxiliar.
26:L PR[3] 500mm/sec FINE ;	Mandamos al robot al punto guardado en el registro de posición
27: PR[13] = UTOOL[1] ;	Reiniciamos el registro de posición

Con estas operaciones conseguimos situarnos en el punto deseado evitando las posibles colisiones con la pieza y usando los puntos guardados en un programa de puntos.

Como muestra de este tipo de programación, podemos observar el diagrama de flujo del programa Fix_rout.



4.1.3 Fig. 1 Diagrama de flujo de un programa de posición

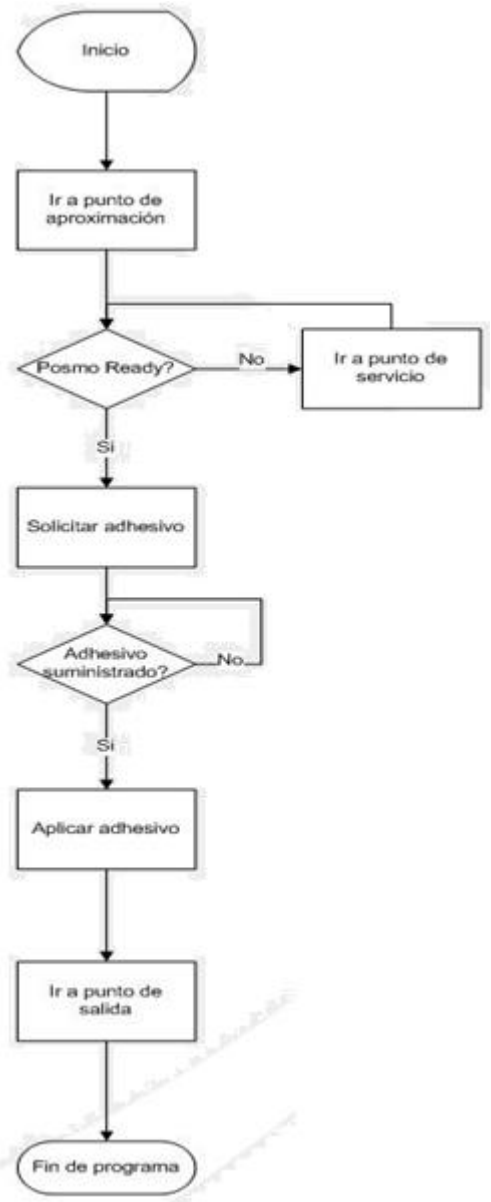
4.2 Descripción de programas

Una vez visto de una forma superficial los diferentes tipos de programas implementados en la aplicación, analizaremos diversos programas que, debido a su relevancia dentro del proceso y la peculiaridad de su programación, merecen un estudio más detallado que el realizado anteriormente.

4.2.1 Programa de aplicación del pegamento

En el programa Bond_app, no es más que un programa de movimiento, realizado mediante programación punto a punto, en el que, además, se puede observar la comunicación entre el PLC y el robot FANUC.

Esta comunicación se hace necesaria al ser el PLC el que controla el motor que dispensa el adhesivo a colocar en el Click Bond.



4.2.1 Fig. 1 Diagrama de flujo del programa BOND_APP

El diagrama de flujo de la página anterior nos muestra el proceso del programa usado para la aplicación del pegamento en el Click Bond. A continuación podemos ver las instrucciones usadas para su realización:

1: !Click bond application ;	
2:	
3: UTOOL_NUM = 1 ;	

4: PAYLOAD[1]	
5: UFRAME_NUM = 1 ;	
6: ;	
7:J P[1:Aproching pos] 100% FINE ;	Punto de aproximación
8: ;	

En estas primeras líneas seleccionamos la herramienta, la carga y el marco de trabajo; después enviamos al robot al punto de aproximación, un punto cercano a la zona de trabajo escogido de forma arbitraria, teniendo en cuenta evitar posibles colisiones en el trayecto desde los puntos de salida de la mesa de Click-Bonds.

Una vez situado el robot en el punto de aproximación, verificamos la disponibilidad del dispensador de adhesivo.

9: LBL[1] ;	
10: DO[42] = ON ;	Petición de apertura del dispensador
11: WAIT DI[42] = ON	Espera de dispensador abierto
12: DO[41] = ON ;	Petición de inicializar Posmo
13: IF DI[41] = ON,JMP LBL[5] ;	Miramos si el Posmo esta operativo
14: ;	
15:L P[3:Bond service pos] 500mm/sec FINE ;	Si el Posmo no esta operativo el robot se retira a la posición de servicio.
16: DO[15] = ON ;	Activamos salida de fallo
17: WAIT DI[41] = ON ;	Esperamos entrada de Posmo OK
18: DO[15] = OFF ;	Desactivamos la salida de fallo
19: JMP LBL[1] ;	
20: ;	
21: LBL[5] ;	
22: ;	

23: !Doser capacity control ;	
24: IF R[4] <= 50,JMP LBL[10] ;	Miramos el contador del dispensador
25: ;	
26: !Doser empty ;	
27: DO[15] = ON ;	Activamos salida de fallo
28: DO[41] = ON ;	Petición de inicializar Posmo
29: ;	
30:L P[3:Bond service pos] 500mm/sec FINE ;	Si el contador supera el máximo de aplicaciones, el robot se retira a la posición de servicio.
31: WAIT DI[41] = OFF ;	Esperamos el permiso para retirar el dispensador
32: ;	
33: !Doser replaced ;	
34: WAIT DI[41] = ON ;	Una vez reemplazado el dispensador el robot espera la señal de Posmo OK
35: DO[41] = OFF ;	Desactivamos inicializar Posmo
36: DO[15] = OFF ;	Desactivamos la señal de fallo
37: JMP LBL[1] ;	

Tanto la inicialización como el reemplazo del dosificador, es realizada por el operario de forma manual, desde la misma consola del robot. Una vez que, tanto el Posmo, como el dosificador se encuentran en estado de funcionamiento, el robot procede a la aplicación del adhesivo en el Clic Bond.

38: ;	
39: LBL[10:Bonding] ;	

40: DO[40] = OFF ; 41: DO[16] = ON ;	Activamos las condiciones iniciales de trabajo del dispensador de adhesivo
42: IF DI[10] = ON,JMP LBL[99] ;	Chequeamos el ciclo en vacío
43: DO[40] = ON ;	Activamos la petición de adhesivo
44: ;	
45: WAIT DI[40] = ON TIMEOUT,LBL[10] ;	Esperamos la señal de fin de proceso
46: DO[40] = OFF ;	Desactivamos la petición de adhesive
47: ;	

En las líneas anteriores podemos ver el proceso de comunicación entre el robot y el PLC.

El PLC recibe la petición del robot y realiza el proceso señalado (activar Posmo); una vez terminado dicho proceso envía al robot la señal indicándole el fin de proceso. Esta señal se resetea desde el PLC al recibir este la desactivación de la señal de petición de adhesivo.

48: LBL[99:END] ;	
49:L P[5:Bonding pos] 400mm/sec FINE ;	El robot se coloca en la posición para aplicar el adhesivo
50: PR[5] = JPOS ;	Cargamos posición actual en registro de posición
51: PR[5,6] = PR[5,6] - 30 ;	Modificamos el eje 6 del registro de posición en 30º.
52:J PR[5] 100% FINE ;	El robot se desplaza a la posición modificada.
53: PR[5] = JPOS ;	Cargamos posición actual en registro de posición
54: PR[5,6] = PR[5,6] + 30 ;	Modificamos el eje 6 del registro de posición en 30º.

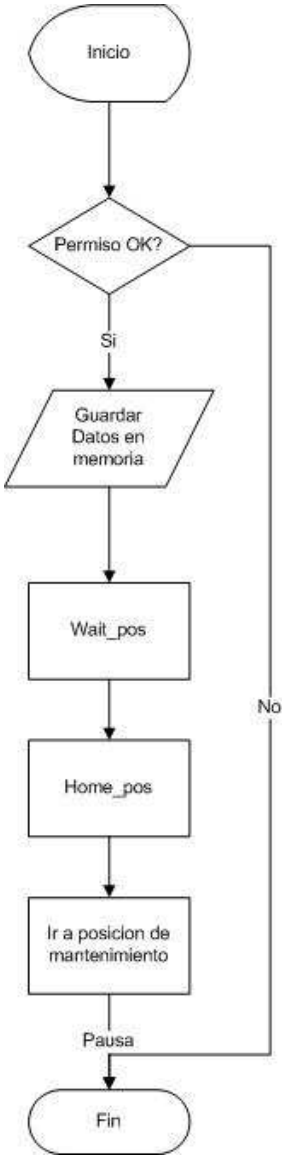
55:J PR[5] 100% FINE ;	El robot se desplaza a la posición modificada.
56: ;	
57: WAIT 0.50(sec) ;	Esperamos 0.5 segundos
58:L P[1:Aproching pos] 400mm/sec CNT50 ;	El robot se desplaza a la posición de aproximamiento
59: DO[42] = OFF ;	Cerramos cobertura del adhesive
60: WAIT 1.00(sec) ;	Esperamos 1 segundo
61: WAIT DI[42] = OFF ;	Esperamos la confirmación de cobertura de adhesive cerrada.

Desde las líneas 48 hasta el final, el robot realiza los movimientos necesarios para que el adhesivo se extienda correcta y uniformemente.

En estas líneas hemos utilizado la instrucción JPOS, que guarda la posición del robot, para después poder modificarla eje a eje en el registro de posición.

4.2.2 Programa de mantenimiento

Como en todo proceso mecánico, pueden surgir imprevistos, fallos eléctricos, etc… Es por esto que se ha previsto un programa para llevar manualmente el robot a mantenimiento, ya que este podría sufrir un fallo en una zona de difícil acceso para los operarios de mantenimiento.



4.2.2 Fig. 1 Diagrama de flujo del programa de mantenimiento.

Debido al tipo de programación que se ha usado para la aplicación, nos encontramos con la problemática de, una vez solucionado el fallo, continuar la aplicación en el punto exacto en el que ocurrió el fallo.

1: !Maintenance position ;	
2: ;	
3: UTOOL_NUM = 1 ;	
4: PAYLOAD[1] ;	
5: UFRAME_NUM = 1 ;	
6: ;	

Como en todos los programas que impliquen movimiento, las primeras líneas las dedicamos a declarar la herramienta, el marco de trabajo y la carga.

7: IF DI[14] = OFF,JMP LBL[99] ;	Miramos si recibimos permiso del PLC para enviar el robot a mantenimiento
8: CALL WAIT_POS ;	Llamamos a la rutina Wait_pos
9: CALL HOME_POS ;	Llamamos a la rutina Home_pos
10: ;	
11:J P[1:Maintenance pos] 100% FINE ;	El robot se desplaza a la posición de mantenimiento
12: DO[14] = ON ;	Informamos al PLC que el robot se encuentra en la zona de mantenimiento
13: PAUSE ;	Pausamos el programa
14: DO[14] = OFF ;	Desactivamos la posición de mantenimiento del robot
16: !Robot home position ;	
17: CALL HOME_POS ;	Llevamos el robot a posición de Home
18: ;	

19: LBL[99:END] ;	Fin de programa.

Una vez se haya realizado el mantenimiento del robot, este habrá guardado los datos necesarios para continuar con el proceso desde el punto exacto donde hubiera surgido el problema; por lo que solo tendremos que reiniciar el programa principal para que el robot continúe las tareas a realizar sin ningún tipo de intervención extra por parte del operario.

4.2 Interconexión PLC-Robot

La comunicación entre el PLC y el robot se realiza mediante el protocolo de comunicación Profibus.

Profibus (Process Field Bus) es un bus de campo industrial utilizado en ámbito de automatización industrial. Se trata de una red abierta, estándar e independiente de cualquier fabricante.

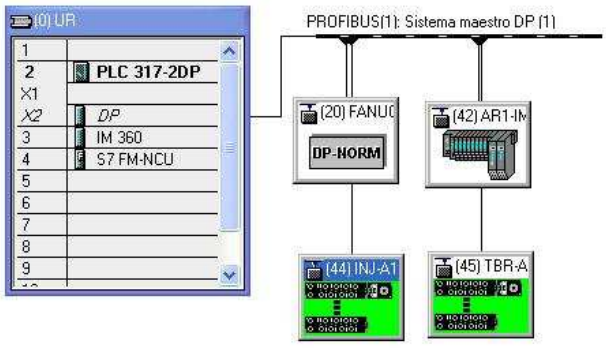
Fue desarrollada en el año 1987 por las empresas alemanas Bosch, Klockner Moller y Siemens. En 1989 la adoptó la norma alemana DIN19245 y fue confirmada como norma europea en 1996 como EN50170.

Este tipo de red trabaja con nodos maestros y nodos esclavos. Los nodos maestros se llaman también activos y los esclavos pasivos.

Además junto con las especificaciones de otros buses de campo se recoge en las normas internacionales IEC61158 e IEC61784.

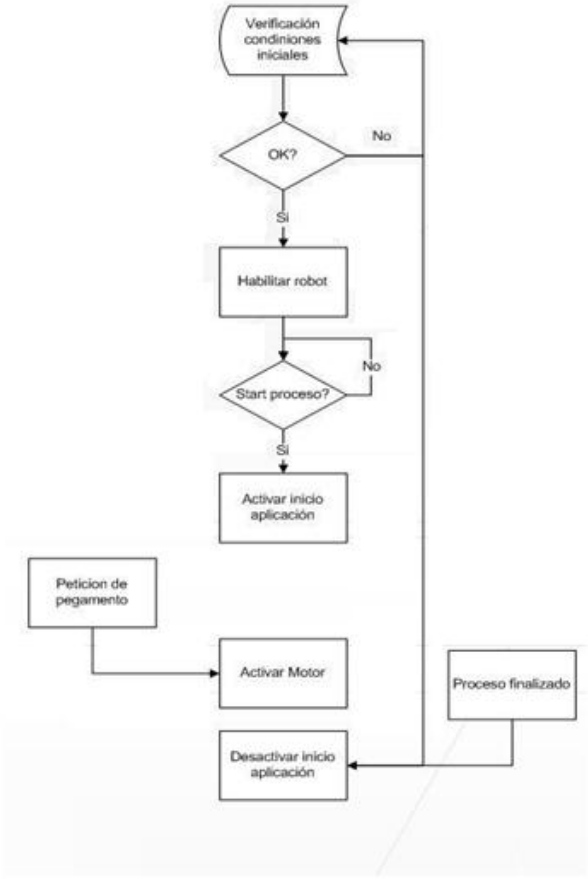
El PLC tiene acceso a las Salidas digitales (DO) del robot, por lo que puede saber el estado de este y actuar en consecuencia, activando las entradas digitales de este.(DI)

Existen varios puntos donde esta comunicación PLC-Robot es bastante estrecha, ya que es el PLC el que habilita el inicio de la aplicación o activa los elementos accesorios al robot; como el motor para la aplicación del pegamento.



4.1.6 Fig. 1. Topología del proceso

Según la figura anterior, podemos observar como a través del PLC, que gestiona las entradas y salidas de todos los elementos implicados en el proceso, el robot obtiene toda la información y los permisos necesarios para desarrollar de manera óptima la aplicación.



4.1.6 Fig. 2. Diagrama de flujo del PLC

En el diagrama 4.1.6 Fig 2, podemos ver como el PLC controla la habilitación del robot cuando se cumplan las condiciones iniciales; pero aun no permitirá el inicio de la aplicación hasta no recibir una señal externa activada por el usuario.

Una vez la aplicación se encuentra en funcionamiento, el PLC controla el cumplimiento de las seguridades incluidas en las condiciones iniciales y la activación del motor del dispensador de adhesivo.

5 CONCLUSIONES

El sistema es capaz de reproducir, de forma eficiente, los procesos que originalmente se llevaban manualmente, con un sistema flexible, una alta repetibilidad y gran precisión.

Introduce notables mejoras en lo referente a tiempo de ciclo disminuyendo desde una media de tiempo de 1 minuto a unos 30 seg por click-bond, reduciendo el tiempo necesario para la finalización de una pieza a la mitad. Con esto se permite encauzar el tiempo hacia otras actividades que favorezcan el aumento de la productividad y que apunten a un mayor crecimiento del servicio.

Siendo la programación del proceso tan compleja, cabe destacar la posibilidad de continuar con el proceso en el mismo punto en que ha sido suspendido en el caso de un fallo externo o una parada de mantenimiento, haciendo más sencillo el uso de la aplicación por parte del usuario normal.

6 TRABAJOS FUTUROS

Con esta aplicación hemos obtenido una gran flexibilidad ante cualquier contingencia o cambio del proceso; pero aun existen campos en los que se podría obtener una mejora sustancial.

Los siguientes trabajos a realizar podrían ser:

- Añadir un sistema de visión para realizar un control de calidad de los Click-bonds insertados.
- Modificación online, vía PLC, de los puntos de inserción de click-bonds.
- Mejora del sistema de herramientas y su programación.

7 Bibliografía

FANUC Robotics; *series (R-J3iC) HANDLING TOOL OPERATOR'S MANUAL*

F. Expósito, I. López y J. Navarro; *Robot Fanuc,* Miniproyecto UPC, 2009

K.S. FU, R.C González, C.S.G. LEE; *Robótica: Control, Detección, Visión e Inteligencia*; McGraw Hill

Angulo, José Ma; *Robótica Practica Tecnología y Aplicaciones*; Ed. Paraninfo

Barrientos, Antonio; *Fundamentos de Robotica;* McGraw Hill

Webs:

Universidad Politécnica de Madrid; *La Industria Aeroespacial 2003.*

http://www.aero.upm.es/departamentos/economia/investiga/informe2003

Clic-bond, Inc

www.**clickbond**.com/

Candelas Rodríguez, Samuel; *Lenguajes de programación de los Robots*

http://www.monografias.com/trabajos3/progrob/progrob.shtml?monosearch

8 Anexos

8.2 Anexo 1

8.1.1 Entradas y salidas digitales del robot

Robot general

INPUTS						**OUTPUTS**					
SIGNAL	**RACK**	**SLOT**	**BIT**	**NAME**		**SIGNAL**	**RACK**	**SLOT**	**BIT**	**NAME**	
UI1	67	1	1	IMSTP*		**UO1**	67	1	1	CMDENBL	
UI2	67	1	2	HOLD*		**UO2**	67	1	2	SYSRDY	
UI3	67	1	3	SFSPD*		**UO3**	67	1	3	PROGRUN	
UI4	67	1	4	CYCLE STOP		**UO4**	67	1	4	PAUSED	
UI5	67	1	5	FAULT RESET		**UO5**	67	1	5	HELD	
UI6	67	1	6	START		**UO6**	67	1	6	FAULT	
UI7	67	1	7			**UO7**	67	1	7	ATPERCH	
UI8	67	1	8	ENABLE*		**UO8**	67	1	8	TPENABLE	
DI9	67	1	9	Cycle start		**DO9**	67	1	9	Release shuttle	
DI10	67	1	10	Dry run		**DO10**	67	1	10	MAIN selected	
DI11	67	1	11	Echo EOS		**DO11**	67	1	11	EOS	
DI12	67	1	12	Click table OK		**DO12**	67	1	12	Release click table	
DI13	67	1	13	Rst click table		**DO13**	67	1	13	Echo Rst click table	
DI14	67	1	14	Maintenance		**DO14**	67	1	14	Maintenance pos	
DI15	67	1	15			**DO15**	67	1	15	Application error	
DI16	67	1	16			**DO16**	67	1	16	In sequence	
DI17	67	1	17	GI1 Model		**DO17**	67	1	17	Collision stop	
DI18	67	1	18			**DO18**	67	1	18	Click table error	
DI19	67	1	19			**DO19**	67	1	19	Auto	
DI20	67	1	20			**DO20**	67	1	20	T1 mode	
DI21	67	1	21	X minus sign		**DO21**	67	1	21	T2 mode	
DI22	67	1	22	GI2 X offset		**DO22**	67	1	22		
DI23	67	1	23			**DO23**	67	1	23		
DI24	67	1	24			**DO24**	67	1	24		
DI25	67	1	25			**DO25**	67	1	25		
DI26	67	1	26	Y minus sign		**DO26**	67	1	26		
DI27	67	1	27	GI3 Y offset		**DO27**	67	1	27		
DI28	67	1	28			**DO28**	67	1	28		
DI29	67	1	29			**DO29**	67	1	29		
DI30	67	1	30			**DO30**	67	1	30		
DI31	67	1	31	Z minus sign		**DO31**	67	1	31		
DI32	67	1	32	GI4 Z offset		**DO32**	67	1	32		
DI33	67	1	33			**DO33**	67	1	33		
DI34	67	1	34			**DO34**	67	1	34		
DI35	67	1	35			**DO35**	67	1	35	Jog+	

Address

PLC Comunication

DI36	67	1	36			**DO36**	67	1	36	Jog-
DI37	67	1	37			**DO37**	67	1	37	Referencing
DI38	67	1	38			**DO38**	67	1	38	Change glue
DI39	67	1	39			**DO39**	67	1	39	New cycle
DI40	67	1	40	EO glueing		**DO40**	67	1	40	Act bond appl
DI41	67	1	41	Bond_ready		**DO41**	67	1	41	initialize posmo
DI42	67	1	42	Bond_cleaned		**DO42**	67	1	42	Activate Posmo cylinder
DI43	67	1	43			**DO43**	67	1	43	In tool change pos
DI44	67	1	44	tool change position		**DO44**	67	1	44	
DI45	67	1	45	tool change done		**DO45**	67	1	45	
DI46	67	1	46			**DO46**	67	1	46	
DI47	67	1	47			**DO47**	67	1	47	
DI48	67	1	48	liquid clean		**DO48**	67	1	48	
DI49	67	1	49			**DO49**	67	1	49	
DI50	67	1	50	Clean tool1		**DO50**	67	1	50	
DI51	67	1	51	Clean tool2		**DO51**	67	1	51	
DI52	67	1	52	Clean tool3		**DO52**	67	1	52	
DI53	67	1	53	Dry tool1		**DO53**	67	1	53	
DI54	67	1	54	Dry tool2		**DO54**	67	1	54	
DI55	67	1	55	Dry tool3		**DO55**	67	1	55	
DI56	67	1	56	tool1		**DO56**	67	1	56	
DI57	67	1	57	tool2		**DO57**	67	1	57	Clean part sensor error
DI58	67	1	58	tool3		**DO58**	67	1	58	Clean grp close error
DI59	67	1	59			**DO59**	67	1	59	Clean grp opn error
DI60	67	1	60			**DO60**	67	1	60	Cyl fwd error
DI61	67	1	61			**DO61**	67	1	61	Cyl bwd error
DI62	67	1	62			**DO62**	67	1	62	Grp close error
DI63	67	1	63			**DO63**	67	1	63	Grp opn error
DI64	67	1	64			**DO64**	67	1	64	Part sensor error

INPUTS **OUTPUTS**

SIGNAL	RACK	SLOT	BIT	NAME		SIGNAL	RACK	SLOT	BIT	NAME
DI65	66	1	1	Cylinder fwd	**PLC Comunication Address**	**DO65**	66	1	1	**NOT AVALIABLE**
DI66	66	1	2	cylinder bwd		**DO66**	66	1	2	**NOT AVALIABLE**
DI67	66	1	3	gripper closed		**DO67**	66	1	3	**NOT AVALIABLE**
DI68	66	1	4	gripper open		**DO68**	66	1	4	**NOT AVALIABLE**
DI69	66	1	5	**NOT AVALIABLE**		**DO69**	66	1	5	Activate cyl
DI70	66	1	6	**NOT AVALIABLE**		**DO70**	66	1	6	Deact cyl
DI71	66	1	7	**NOT AVALIABLE**		**DO71**	66	1	7	Close gripper
DI72	66	1	8	**NOT AVALIABLE**		**DO72**	66	1	8	Open gripper

DI73	66	1	9	tool in clean gripper		**DO73**	66	1	9	**NOT AVALIABLE**
DI74	66	1	10	Part sensor		**DO74**	66	1	10	**NOT AVALIABLE**
DI75	66	1	11	Open Clean grip		**DO75**	66	1	11	**NOT AVALIABLE**
DI76	66	1	12	Close clean grip		**DO76**	66	1	12	**NOT AVALIABLE**
DI77	66	1	13	**NOT AVALIABLE**		**DO77**	66	1	13	Close clean gripp
DI78	66	1	14	**NOT AVALIABLE**		**DO78**	66	1	14	Open clean gripper
DI79	66	1	15	**NOT AVALIABLE**		**DO79**	66	1	15	
DI80	66	1	16	**NOT AVALIABLE**		**DO80**	66	1	16	
DI81	66	1	16	pres sensor		**DO81**	66	1	17	**NOT AVALIABLE**
DI82	66	1	16			**DO82**	66	1	18	**NOT AVALIABLE**
DI83	66	1	16			**DO83**	66	1	19	**NOT AVALIABLE**
DI84	66	1	16			**DO84**	66	1	20	**NOT AVALIABLE**
DI85	66	1	16	**NOT AVALIABLE**		**DO85**	66	1	21	
DI86	66	1	16	**NOT AVALIABLE**		**DO86**	66	1	22	
DI87	66	1	16	**NOT AVALIABLE**		**DO87**	66	1	23	
DI88	66	1	16	**NOT AVALIABLE**		**DO88**	66	1	24	

8.2 Anexo 2

8.2.1 ABRA_ROU

```
/PROG  ABRA_ROU
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Clean all positi";
PROG_SIZE  = 2360;
CREATE                = DATE 08-11-11
TIME 19:39:23;
MODIFIED    = DATE 08-12-16  TIME
16:47:21;
FILE_NAME  = ;
VERSION     = 0;
LINE_COUNT          = 127;
MEMORY_SIZE       = 2732;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Working point calculation ;
   2:   ;
   3:  UTOOL_NUM = 4 ;
   4:  PAYLOAD[2] ;
   5:  UFRAME_NUM = 2 ;
   6:   ;
   7:  DO[9] = OFF ;
   8:  WAIT DI[12] = ON ;
   9:  R[8] = 10 ;
  10:  PR[12] =  UTOOL[2] ;
  11:  PR[13] =  UTOOL[3] ;
  12:  IF R[13] = 1,JMP LBL[99]  ;
  13:   ;
  14:  LBL[100:Start] ;
  15:  IF R[13] = 1,JMP LBL[99]  ;
  16:  IF DI[10] = ON,JMP LBL[101]  ;
  17:  IF DI[73] = OFF OR DI[74] =
OFF,JMP LBL[500]  ;
  18:   ;
  19:  LBL[101] ;
  20:  !Abrasive section ;
  21:  PR[12] =  UTOOL[2] ;
  22:  PR[13] =  UTOOL[2] ;
  23:   ;
  24:  !Distance calculation ;
  25:  PR[GP1:13,10] = PR[13,10] - 0 ;
  26:  PR[GP1:13,11] = PR[13,11] + 0 ;
  27:  PR[GP1:13,12] = PR[13,12] + 400 ;
  28:   ;
  29:  !Aproching point ;
  30:  CALL OPEN_CLN ;
  31:   UTOOL[4] = PR[13] ;
  32:  UTOOL_NUM = 4 ;
33:L PR[3] 700mm/sec FINE  ;
  34:   ;
  35:  !Fixing point ;
  36:   UTOOL[4] = PR[12] ;
37:L PR[3] 500mm/sec FINE  ;
  38:   ;
  39:  IF DI[10] = ON,JMP LBL[102]  ;
  40:   ;
  41:  LBL[200] ;
  42:  PR[GP1:13,12] = PR[13,12] - 1 ;
  43:  !Aproching point ;
  44:   UTOOL[4] = PR[13] ;
  45:  UTOOL_NUM = 4 ;
46:L PR[3] 700mm/sec FINE  ;
  47:   ;
  48:  IF DI[10] = ON,JMP LBL[102]  ;
  49:  IF DI[81] = OFF,JMP LBL[200]  ;
  50:   ;
  51:  LBL[102] ;
  52:  PR[5] = JPOS ;
  53:  PR[5,6] = PR[5,6] - 90 ;
54:J PR[5] 100% FINE  ;
  55:  PR[5,6] = PR[5,6] + 90 ;
56:J PR[5] 100% FINE  ;
  57:  R[8] = R[8] - 1 ;
  58:   ;
  59:  IF R[8] <> 0,JMP LBL[102]  ;
  60:   ;
  61:  LBL[201] ;
  62:   ;
```

```
 63:   UTOOL[4] = PR[13] ;
 64:  !Distance calculation ;
 65:  PR[GP1:13,10] = PR[13,10] - 0 ;
 66:  PR[GP1:13,11] = PR[13,11] + 0 ;
 67:  PR[GP1:13,12] = PR[13,12] + 400 ;
 68:   ;
 69:  LBL[197] ;
 70:  !Return point ;
 71:L PR[3] 700mm/sec FINE  ;
 72:   ;
 73:  LBL[198] ;
 74:  !Clean point routine ;
 75:  CALL CLOS_CLN ;
 76:  IF DI[10] = ON,JMP LBL[298]  ;
 77:  IF DI[73] = ON,JMP LBL[298]  ;
 78:  JMP LBL[503]  ;
 79:   ;
 80:  LBL[298] ;
 81:  !Fixing point ;
 82:   UTOOL[4] = PR[14] ;
 83:L PR[3] 500mm/sec FINE  ;
 84:  IF DI[10] = ON,JMP LBL[203]  ;
 85:   ;
 86:  LBL[251] ;
 87:  PR[GP1:13,12] = PR[13,12] - 1 ;
 88:  !Aproching point ;
 89:   UTOOL[4] = PR[13] ;
 90:  UTOOL_NUM = 4 ;
 91:L PR[3] 700mm/sec FINE  ;
 92:  IF DI[10] = ON,JMP LBL[203]  ;
 93:  IF DI[84] = OFF,JMP LBL[251]  ;
 94:   ;
 95:  LBL[203] ;
 96:  PR[5] = JPOS ;
 97:  PR[5,6] = PR[5,6] - 90 ;
 98:J PR[5] 100% FINE  ;
 99:  PR[5,6] = PR[5,6] + 90 ;
100:J PR[5] 100% FINE  ;
101:  !Return point ;
102:  PR[13] =  UTOOL[2] ;
103:   ;
104:  !Distance calculation ;
105:  PR[GP1:13,10] = PR[13,10] - 0 ;
106:  PR[GP1:13,11] = PR[13,11] + 0 ;
107:  PR[GP1:13,12] = PR[13,12] + 400 ;
108:  !Return point ;
109:   UTOOL[4] = PR[13] ;
110:L PR[3] 700mm/sec FINE  ;
111:  DO[16] = OFF ;
112:   ;
113:  JMP LBL[99]  ;
114:   ;
115:  LBL[500:Alarms] ;
116:  IF DI[10] = ON,JMP LBL[10]  ;
117:  DO[22] = ON ;
118:   ;
119:  UALM[4] ;
120:   ;
121:  CALL MAINTEN ;
122:  WAIT DI[73] = ON AND DI[74] =
ON ;
123:  DO[22] = OFF ;
124:  DO[15] = OFF ;
125:  JMP LBL[100]  ;
126:   ;
127:  LBL[99] ;
/POS
/END
```

8.2.2 ACT_CYL

```
/PROG  ACT_CYL   MACRO
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Activate cylindr";
PROG_SIZE  = 988;
CREATE                = DATE 07-11-21
TIME 16:22:04;
MODIFIED    = DATE 08-11-13  TIME
20:20:27;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 53;
MEMORY_SIZE       = 1384;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = *,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Activate cylinder ;
   2:  IF DI[65] = OFF AND DI[66] =
ON,JMP LBL[100]  ;
   3:  IF DI[65] = ON AND DI[66] =
OFF,JMP LBL[200]  ;
   4:  IF DI[65] = OFF AND DI[66] = OFF
AND DI[73] = ON,JMP LBL[200]  ;
   5:   ;
   6:  LBL[100:Activate cylinde] ;
   7:  DO[70] = OFF ;
   8:  DO[69] = ON ;
   9:  JMP LBL[101]  ;
  10:  WAIT DI[66] = OFF
TIMEOUT,LBL[10] ;
  11:  LBL[101] ;
  12:  WAIT      1.00(sec) ;
  13:  DO[69] = OFF ;
  14:  JMP LBL[99]  ;
  15:   ;
  16:  LBL[200:Deactivate cylin] ;
  17:  DO[69] = OFF ;
  18:  DO[70] = ON ;
  19:  WAIT DI[66] = ON AND DI[65] =
OFF TIMEOUT,LBL[20] ;
  20:  WAIT      1.00(sec) ;
  21:  DO[70] = OFF ;
  22:  JMP LBL[99]  ;
  23:   ;
  24:  LBL[99] ;
  25:  END ;
  26:   ;
  27:  LBL[10:Error handler] ;
  28:  IF DI[66] = OFF,JMP LBL[100]  ;
  29:  DO[15] = ON ;
  30:  DO[60] = ON ;
  31:  DO[22] = ON ;
  32:   ;
  33:  UALM[8] ;
  34:   ;
  35:  CALL MAINTEN ;
  36:  WAIT DI[66] = OFF ;
  37:  DO[60] = OFF ;
  38:  DO[22] = OFF ;
  39:  JMP LBL[99]  ;
  40:   ;
  41:  LBL[20] ;
  42:  IF DI[66] = ON AND DI[65] =
OFF,JMP LBL[99]  ;
  43:  DO[61] = ON ;
  44:  DO[22] = ON ;
  45:   ;
  46:  UALM[8] ;
  47:   ;
  48:  CALL MAINTEN ;
  49:  WAIT DI[66] = ON AND DI[65] =
OFF ;
  50:  DO[22] = OFF ;
  51:  DO[61] = OFF ;
  52:  DO[15] = OFF ;
  53:   ;
/POS
/END
```

8.2.3 BOND_APP

```
/PROG  BOND_APP
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Bonding aplicati";
PROG_SIZE  = 1416;
CREATE                = DATE 07-11-27
TIME 09:27:27;
MODIFIED    = DATE 08-12-16  TIME
21:30:07;
FILE_NAME  = ;
VERSION     = 0;
LINE_COUNT          = 63;
MEMORY_SIZE       = 1764;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE      = 0,
     BUSY_LAMP_OFF       = 0,
     ABORT_REQUEST       = 0,
     PAUSE_REQUEST       = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/MN
   1:  !Click bond application ;
   2:   ;
   3:  UTOOL_NUM = 1 ;
   4:  PAYLOAD[1] ;
   5:  UFRAME_NUM = 1 ;
   6:   ;
   7:J P[1:Aproching pos] 100% FINE  ;
   8:   ;
   9:  LBL[1] ;
  10:  DO[42] = ON ;
  11:  WAIT DI[42] = ON ;
  12:  DO[41] = ON ;
  13:  IF DI[41] = ON,JMP LBL[5]  ;
  14:   ;
  15:L P[3:Bond service pos] 500mm/sec
FINE  ;
  16:  DO[15] = ON ;
  17:  WAIT DI[41] = ON ;
  18:  DO[15] = OFF ;
  19:  JMP LBL[1]  ;
  20:   ;
  21:  LBL[5] ;
  22:   ;
  23:  !Doser capacity control ;
  24:  IF R[4] <= 50,JMP LBL[10]  ;
  25:   ;
  26:  !Doser empty ;
  27:  DO[15] = ON ;
  28:  DO[41] = ON ;
  29:   ;
  30:L P[3:Bond service pos] 500mm/sec
FINE  ;
  31:  WAIT DI[41] = OFF ;
  32:   ;
  33:  !Doser replaced ;
  34:  WAIT DI[41] = ON ;
  35:  DO[41] = OFF ;
  36:  DO[15] = OFF ;
  37:  JMP LBL[1]  ;
  38:   ;
  39:  LBL[10:Bonding] ;
  40:  DO[40] = OFF ;
  41:  DO[16] = ON ;
  42:  IF DI[10] = ON,JMP LBL[99]  ;
  43:  DO[40] = ON ;
  44:   ;
  45:  WAIT DI[40] = ON
TIMEOUT,LBL[10] ;
  46:  DO[40] = OFF ;
  47:   ;
  48:  LBL[99:END] ;
  49:L P[3:Bond service pos] 400mm/sec
FINE  ;
  50:  PR[5] = JPOS ;
  51:  PR[5,6] = PR[5,6] - 30 ;
  52:J PR[5] 100% FINE  ;
  53:  PR[5] = JPOS ;
  54:  PR[5,6] = PR[5,6] + 30 ;
  55:J PR[5] 100% FINE  ;
  56:   ;
  57:  WAIT      0.50(sec) ;
  58:L P[1:Aproching pos] 400mm/sec
CNT50  ;
  59:  DO[42] = OFF ;
```

```
  60:  WAIT      1.00(sec) ;
  61:  WAIT DI[42] = OFF ;
  62:  DO[42] = OFF ;
  63:   ;
/END
```

8.2.4 CLN_ROUT

```
/PROG  CLN_ROUT
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Clean routine";
PROG_SIZE   = 2456;
CREATE                = DATE 08-02-18
TIME 16:46:25;
MODIFIED    = DATE 08-12-16  TIME
16:11:07;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 133;
MEMORY_SIZE       = 2932;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Working point calculation ;
   2:   ;
   3:  UTOOL_NUM = 4 ;
   4:  PAYLOAD[2] ;
   5:  UFRAME_NUM = 2 ;
   6:   ;
   7:  DO[9] = OFF ;
   8:  WAIT DI[12] = ON ;
   9:  R[8] = 1 ;
  10:  PR[12] =  UTOOL[2] ;
  11:  PR[13] =  UTOOL[2] ;
  12:  IF R[13] = 0,JMP LBL[99]  ;
  13:   ;
  14:  LBL[100:Start] ;
  15:  IF DI[10] = ON,JMP LBL[101]  ;
  16:  IF DI[73] = OFF OR DI[74] =
OFF,JMP LBL[500]  ;
  17:   ;
  18:   ;
  19:  LBL[101] ;
  20:  !Wash section ;
  21:  PR[12] =  UTOOL[2] ;
  22:  PR[13] =  UTOOL[2] ;
  23:   ;
  24:   UTOOL[4] = PR[13] ;
  25:  !Distance calculation ;
  26:  PR[GP1:13,10] = PR[13,10] - 0 ;
  27:  PR[GP1:13,11] = PR[13,11] + 0 ;
  28:  PR[GP1:13,12] = PR[13,12] + 200 ;
  29:   ;
  30:  !Aproching point ;
  31:  CALL OPEN_CLN ;
  32:   UTOOL[4] = PR[13] ;
  33:  UTOOL_NUM = 4 ;
34:L PR[3] 1200mm/sec FINE  ;
  35:   ;
  36:  !Wash point ;
  37:   UTOOL[4] = PR[12] ;
38:L PR[3] 500mm/sec FINE  ;
  39:   ;
  40:  IF DI[10] = ON,JMP LBL[102]  ;
  41:   ;
  42:  LBL[200] ;
  43:  PR[GP1:13,12] = PR[13,12] - 0.5 ;
  44:  !Aproaching point ;
  45:   UTOOL[4] = PR[13] ;
  46:  UTOOL_NUM = 4 ;
47:L PR[3] 500mm/sec FINE  ;
  48:   ;
  49:  IF DI[10] = ON,JMP LBL[102]  ;
  50:  IF DI[84] = OFF,JMP LBL[200]  ;
  51:   ;
  52:  LBL[102] ;
  53:  PR[5] = JPOS ;
  54:  PR[5,6] = PR[5,6] - 90 ;
55:J PR[5] 100% FINE  ;
  56:  PR[5,6] = PR[5,6] + 90 ;
57:J PR[5] 100% FINE  ;
  58:  R[8] = R[8] - 1 ;
  59:   ;
  60:  IF R[8] <> 0,JMP LBL[102]  ;
  61:   ;
  62:  LBL[201:Clean gripper] ;
```

```
 63:  !Clean/Dry section ;
 64:  PR[13] =  UTOOL[3] ;
 65:   ;
 66:   UTOOL[4] = PR[13] ;
 67:  !Distance calculation ;
 68:  PR[GP1:13,10] = PR[13,10] - 0 ;
 69:  PR[GP1:13,11] = PR[13,11] + 0 ;
 70:  PR[GP1:13,12] = PR[13,12] + 200 ;
 71:   ;
 72:  LBL[197] ;
 73:  !Return point ;
 74:   UTOOL[4] = PR[13] ;
 75:  UTOOL_NUM = 4 ;
 76:L PR[3] 1200mm/sec FINE  ;
 77:   ;
 78:  LBL[198] ;
 79:  !Cleaning/Drying point ;
 80:  CALL CLOS_CLN ;
 81:  IF DI[10] = ON,JMP LBL[298]  ;
 82:  IF DI[73] = ON,JMP LBL[298]  ;
 83:  JMP LBL[503]  ;
 84:   ;
 85:  LBL[298] ;
 86:  !Fixing point ;
 87:   UTOOL[4] = PR[14] ;
 88:L PR[3] 500mm/sec FINE  ;
 89:  PR[13] = PR[14] ;
 90:  IF DI[10] = ON,JMP LBL[203]  ;
 91:   ;
 92:  LBL[250] ;
 93:  PR[GP1:13,12] = PR[13,12] - 2 ;
 94:  !Aproaching point ;
 95:   UTOOL[4] = PR[13] ;
 96:  UTOOL_NUM = 4 ;
 97:L PR[3] 500mm/sec FINE  ;
 98:  IF DI[10] = ON,JMP LBL[203]  ;
 99:  IF DI[84] = OFF,JMP LBL[250]  ;
100:   ;
101:  LBL[203] ;
102:  PR[5] = JPOS ;
103:  PR[5,6] = PR[5,6] - 90 ;
104:J PR[5] 100% FINE  ;
105:  PR[5,6] = PR[5,6] + 90 ;
106:J PR[5] 100% FINE  ;
107:  !Return point ;
108:  PR[13] =  UTOOL[2] ;
109:   ;
110:  !Distance calculation ;
111:  PR[GP1:13,10] = PR[13,10] + 0 ;
112:  PR[GP1:13,11] = PR[13,11] + 0 ;
113:  PR[GP1:13,12] = PR[13,12] + 200
;
114:  !Return point ;
115:   UTOOL[4] = PR[13] ;
116:L PR[3] 1500mm/sec FINE  ;
117:  DO[16] = OFF ;
118:   ;
119:  JMP LBL[99]  ;
120:   ;
121:  LBL[500:Alarms] ;
122:  DO[15] = ON ;
123:  DO[22] = ON ;
124:   ;
125:  UALM[4] ;
126:   ;
127:  CALL MAINTEN ;
128:  WAIT DI[73] = ON AND DI[74] =
ON ;
129:  DO[22] = OFF ;
130:  DO[15] = OFF ;
131:  JMP LBL[100]  ;
132:   ;
133:  LBL[99] ;
/POS
/END
```

8.2.5 CLOS_CLN

```
/PROG  CLOS_CLN MACRO
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Close gripper";
PROG_SIZE  = 746;
CREATE                = DATE 08-03-12
TIME 16:23:08;
MODIFIED    = DATE 08-11-12  TIME
16:37:24;
FILE_NAME = ;
VERSION     = 0;
LINE_COUNT          = 43;
MEMORY_SIZE       = 1182;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE      = 0,
     BUSY_LAMP_OFF       = 0,
     ABORT_REQUEST       = 0,
     PAUSE_REQUEST       = 0;
DEFAULT_GROUP  = *,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Close clean gripper ;
   2:   ;
   3:  IF DI[76] = ON,JMP LBL[1]  ;
   4:   ;
   5:  DO[78] = OFF ;
   6:  DO[77] = ON ;
   7:   ;
   8:  LBL[1] ;
   9:  WAIT DI[75] = OFF AND DI[76] =
ON TIMEOUT,LBL[10] ;
  10:  DO[77] = OFF ;
  11:   ;
  12:  END ;
  13:   ;
  14:  LBL[10:Error handler] ;
  15:  DO[15] = ON ;
  16:   ;
  17:  IF DI[76] = OFF,JMP LBL[20]  ;
  18:  DO[59] = ON ;
  19:  DO[22] = ON ;
  20:   ;
  21:  UALM[3] ;
  22:   ;
  23:  CALL MAINTEN ;
  24:  WAIT DI[76] = OFF ;
  25:  DO[58] = OFF ;
  26:  DO[22] = OFF ;
  27:  LBL[20] ;
  28:  IF DI[76] = ON,JMP LBL[99]  ;
  29:  DO[58] = ON ;
  30:  DO[22] = ON ;
  31:   ;
  32:  UALM[3] ;
  33:   ;
  34:  CALL MAINTEN ;
  35:  WAIT DI[76] = ON ;
  36:  DO[58] = OFF ;
  37:  DO[22] = OFF ;
  38:  JMP LBL[99]  ;
  39:   ;
  40:  LBL[99:END] ;
  41:  DO[15] = OFF ;
  42:   ;
  43:  JMP LBL[1]  ;
/POS
/END
```

8.2.6 CLOS_GRP

```
/PROG  CLOS_GRP  MACRO
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Close gripper";
PROG_SIZE  = 698;
CREATE                = DATE 07-11-19
TIME 17:40:20;
MODIFIED    = DATE 08-11-12  TIME
16:39:29;
FILE_NAME  = ;
VERSION     = 0;
LINE_COUNT          = 43;
MEMORY_SIZE       = 1134;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE      = 0,
     BUSY_LAMP_OFF       = 0,
     ABORT_REQUEST       = 0,
     PAUSE_REQUEST       = 0;
DEFAULT_GROUP  = *,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/MN
   1:  !Close gripper ;
   2:   ;
   3:  IF DI[67] = ON,JMP LBL[1]  ;
   4:   ;
   5:  DO[72] = OFF ;
   6:  DO[71] = ON ;
   7:   ;
   8:  LBL[1] ;
   9:  WAIT DI[68] = OFF AND DI[67] =
ON TIMEOUT,LBL[10] ;
  10:   ;
  11:  DO[71] = OFF ;
  12:  END ;
  13:   ;
  14:  LBL[10:Error handler] ;
  15:  DO[15] = ON ;
  16:  DO[22] = ON ;
  17:  IF DI[67] = OFF,JMP LBL[20]  ;
  18:  DO[63] = ON ;
  19:   ;
  20:  UALM[4] ;
  21:   ;
  22:  CALL MAINTEN ;
  23:  WAIT DI[67] = OFF ;
  24:  DO[63] = OFF ;
  25:  DO[22] = OFF ;
  26:   ;
  27:  LBL[20] ;
  28:  IF DI[67] = ON,JMP LBL[99]  ;
  29:  DO[62] = ON ;
  30:  DO[22] = ON ;
  31:   ;
  32:  UALM[4] ;
  33:   ;
  34:  CALL MAINTEN ;
  35:  WAIT DI[67] = ON ;
  36:  DO[62] = OFF ;
  37:  DO[22] = OFF ;
  38:  JMP LBL[99]  ;
  39:   ;
  40:  LBL[99:END] ;
  41:  DO[15] = OFF ;
  42:   ;
  43:  JMP LBL[1]  ;
/POS
/END
```

8.2.7 FIX_ROUT

```
/PROG  FIX_ROUT
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Fix routine";
PROG_SIZE  = 1200;
CREATE                = DATE 07-11-29
TIME 10:56:11;
MODIFIED    = DATE 08-12-16  TIME
18:03:29;
FILE_NAME  = ;
VERSION     = 0;
LINE_COUNT          = 66;
MEMORY_SIZE       = 1680;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE      = 0,
     BUSY_LAMP_OFF       = 0,
     ABORT_REQUEST       = 0,
     PAUSE_REQUEST       = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/MN
   1:  !Working point calculation ;
   2:   ;
   3:  UTOOL_NUM = 1 ;
   4:  PAYLOAD[1] ;
   5:  UFRAME_NUM = 2 ;
   6:   ;
   7:  DO[9] = OFF ;
   8:   ;
   9:   ;
  10:  LBL[10:Start] ;
  11:  !Dry section ;
  12:  PR[11] =  UTOOL[1] ;
  13:  PR[13] =  UTOOL[1] ;
  14:   ;
  15:  !Distance calculation ;
  16:  PR[GP1:13,10] = PR[13,10] - 0 ;
  17:  PR[GP1:13,11] = PR[13,11] + 0 ;
  18:  PR[GP1:13,12] = PR[13,12] + 140 ;
  19:  !Aproching point ;
  20:   UTOOL[4] = PR[13] ;
  21:  UTOOL_NUM = 4 ;
22:L PR[3] 1000mm/sec FINE  ;
  23:   ;
  24:  !Fixing point ;
  25:   UTOOL[4] = PR[11] ;
26:L PR[3] 500mm/sec FINE  ;
  27:  PR[13] =  UTOOL[1] ;
  28:   ;
  29:  IF DI[10] = ON,JMP LBL[201]  ;
  30:   ;
  31:  LBL[200] ;
  32:  PR[GP1:13,12] = PR[13,12] - 5 ;
  33:  !Aproching point ;
  34:   UTOOL[4] = PR[13] ;
  35:  UTOOL_NUM = 4 ;
36:L PR[3] 1000mm/sec FINE  ;
  37:  IF DI[81] = OFF,JMP LBL[200]  ;
  38:   ;
  39:  LBL[201] ;
  40:  CALL OPEN_GRP ;
  41:  CALL ACT_CYL ;
  42:  WAIT      1.50(sec) ;
  43:  R[10] = 0 ;
  44:  !Return point ;
  45:  PR[11] =  UTOOL[1] ;
  46:  PR[13] =  UTOOL[1] ;
  47:   ;
  48:  !Distance calculation ;
  49:  PR[GP1:13,10] = PR[13,10] - 0 ;
  50:  PR[GP1:13,11] = PR[13,11] + 0 ;
  51:  PR[GP1:13,12] = PR[13,12] + 140 ;
  52:  !Aproching point ;
  53:   UTOOL[4] = PR[13] ;
  54:  UTOOL_NUM = 4 ;
55:L PR[3] 500mm/sec FINE  ;
  56:   ;
  57:  CALL ACT_CYL ;
  58:   ;
  59:  DO[16] = OFF ;
  60:   ;
  61:   ;
  62:  UTOOL_NUM = 1 ;
  63:  PAYLOAD[1] ;
  64:  UFRAME_NUM = 2 ;
```

```
  65:   ;
  66:  LBL[99] ;
/POS
/END
```

8.2.8 GET_ABRA

```
/PROG  GET_ABRA
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "";
PROG_SIZE   = 3078;
CREATE                = DATE 08-10-28
TIME 11:22:06;
MODIFIED    = DATE 08-12-16  TIME
19:27:14;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 152;
MEMORY_SIZE       = 3634;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Get wash & dry tools from table ;
   2:  UTOOL_NUM = 4 ;
   3:  PAYLOAD[2] ;
   4:  UFRAME_NUM = 3 ;
   5:   ;
   6:  PR[11] =  UTOOL[1] ;
   7:  PR[12] =  UTOOL[2] ;
   8:  PR[14] =  UTOOL[3] ;
   9:   UTOOL[4] = PR[12] ;
  10:   ;
  11:  WAIT DI[12] = ON ;
  12:  IF DI[10] = ON,JMP LBL[2]  ;
  13:  LBL[10:Get clean tool] ;
  14:  IF R[11] <> 1 AND DI[73] =
OFF,JMP LBL[19]  ;
  15:   ;
  16:  LBL[2] ;
  17:  IF R[11] > R[15] AND DI[73] =
ON,JMP LBL[99]  ;
  18:  IF R[11] > R[15] AND DI[10] =
ON,JMP LBL[99]  ;
  19:  IF R[11] <> 1 AND DI[10] =
ON,JMP LBL[200]  ;
  20:  IF R[11] <> 1 AND DI[73] =
OFF,JMP LBL[18]  ;
  21:   ;
  22:  LBL[200] ;
  23:   ;
  24:  PR[13] =  UTOOL[3] ;
  25:   ;
  26:  CALL CLOS_CLN ;
  27:   ;
  28:   UTOOL[4] = PR[13] ;
  29:   ;
  30:  R[9] = 73 ;
  31:   ;
  32:  LBL[14] ;
  33:  IF DI[53] = ON,JMP LBL[15]  ;
  34:  IF DI[54] = ON,JMP LBL[16]  ;
  35:  IF DI[10] = ON,JMP LBL[15]  ;
  36:   ;
  37:  UALM[5] ;
  38:  WAIT DI[53] = ON OR DI[54] =
ON OR DI[55] = ON ;
  39:  JMP LBL[14]  ;
  40:   ;
  41:  LBL[15:Clean tool1] ;
  42:  PR[4] = P[11] ;
  43:  R[8] = 53 ;
  44:  JMP LBL[100]  ;
  45:   ;
  46:  LBL[16:Clean tool2] ;
  47:  PR[4] = P[12] ;
  48:  R[8] = 54 ;
  49:  JMP LBL[100]  ;
  50:   ;
  51:  LBL[18:Clean tool error] ;
  52:  DO[57] = ON ;
  53:  DO[15] = ON ;
  54:   ;
  55:  WAIT DI[73] = ON ;
  56:  DO[15] = OFF ;
```

```
 57:  DO[57] = OFF ;
 58:  JMP LBL[10]  ;
 59:   ;
 60:  LBL[19:Reject tool] ;
 61:  CALL REJECT ;
 62:  JMP LBL[10]  ;
 63:   ;
 64:  LBL[20:Get abrasive too] ;
 65:  IF DI[74] = ON,JMP LBL[110]  ;
 66:   ;
 67:  LBL[25] ;
 68:  R[9] = 74 ;
 69:   ;
 70:  CALL OPEN_CLN ;
 71:  CALL OPEN_GRP ;
 72:  PR[13] =  UTOOL[2] ;
 73:  IF DI[56] = ON,JMP LBL[11]  ;
 74:  IF DI[10] = ON,JMP LBL[11]  ;
 75:  DO[22] = ON ;
 76:   ;
 77:  UALM[5] ;
 78:   ;
 79:  CALL MAINTEN ;
 80:  WAIT DI[56] = ON ;
 81:  DO[22] = OFF ;
 82:  JMP LBL[10]  ;
 83:   ;
 84:  LBL[11:Abrasive tool] ;
 85:  PR[4] = P[1] ;
 86:  R[8] = 56 ;
 87:  JMP LBL[100]  ;
 88:   ;
 89:   ;
 90:  LBL[100:Start] ;
 91:  PR[13,12] = PR[13,12] + 300 ;
 92:   UTOOL[4] = PR[13] ;
 93:L PR[4] 1500mm/sec FINE  ;
 94:   ;
 95:  PR[13,12] = PR[13,12] - 300 ;
 96:   UTOOL[4] = PR[13] ;
 97:L PR[4] 100mm/sec FINE  ;
 98:   ;
 99:  IF DI[10] = ON,JMP LBL[101]  ;
100: Undefined instruction;
101:  WAIT DI[R[9]] = ON AND
DI[R[9]] = ON TIMEOUT,LBL[500] ;
102:   ;
103:  LBL[101:Sensor alarm ret] ;
104:   ;
105:  IF R[9] = 73,JMP LBL[105]  ;
106:   ;
107:  LBL[104] ;
108:   ;
109:  CALL CLOS_GRP ;
110:   ;
111:  LBL[105] ;
112:   ;
113:  PR[13,12] = PR[13,12] + 300 ;
114:   UTOOL[4] = PR[13] ;
115:L PR[4] 500mm/sec FINE  ;
116:  IF DI[10] = ON,JMP LBL[107]  ;
117:  WAIT DI[R[8]] = OFF AND
DI[R[9]] = ON TIMEOUT,LBL[500] ;
118:   ;
119:  LBL[107] ;
120:  IF DI[73] = ON AND DI[74] =
ON,JMP LBL[99]  ;
121:  IF R[9] = 74 AND DI[10] =
ON,JMP LBL[99]  ;
122:  JMP LBL[20]  ;
123:   ;
124:  LBL[108] ;
125:  WAIT DI[74] = ON AND DI[R[8]]
= OFF ;
126:  JMP LBL[99]  ;
127:   ;
128:  LBL[99:END] ;
129:  R[8] = 0 ;
130:  R[9] = 0 ;
131:  END ;
132:   ;
133:  LBL[500:Sensor alarm] ;
134:  DO[22] = ON ;
135:   ;
136:  UALM[4] ;
137:   ;
138:  DO[22] = OFF ;
139:  JMP LBL[101]  ;
140:   ;
```

```
 141:  !Tool adquire points ;
 142:  !Abrasive tool point ;
 143:  UTOOL_NUM = 2 ;
 144:  UFRAME_NUM = 3 ;
 145:L P[1:Abrasive Tool1] 100mm/sec
FINE  ;
 146:   ;
 147:  !Clean tool point ;
 148:  UTOOL_NUM = 3 ;
 149:  UFRAME_NUM = 3 ;
 150:L P[11:Tool clean1] 100mm/sec FINE
;
 151:L P[12:Tool clean2] 100mm/sec FINE
;
 152:L P[13:Tool clean3] 100mm/sec FINE
;
```

8.2.9 GET_WASH

```
/PROG  GET_WASH
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "";
PROG_SIZE  = 3250;
CREATE                = DATE 08-10-29
TIME 09:48:26;
MODIFIED    = DATE 08-12-16  TIME
19:25:09;
FILE_NAME  = ;
VERSION     = 0;
LINE_COUNT          = 155;
MEMORY_SIZE       = 3794;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE      = 0,
     BUSY_LAMP_OFF       = 0,
     ABORT_REQUEST       = 0,
     PAUSE_REQUEST       = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Get wash & dry tools from table ;
   2:  UTOOL_NUM = 4 ;
   3:  PAYLOAD[2] ;
   4:  UFRAME_NUM = 3 ;
   5:   ;
   6:  PR[11] =  UTOOL[1] ;
   7:  PR[12] =  UTOOL[2] ;
   8:  PR[14] =  UTOOL[3] ;
   9:   UTOOL[4] = PR[12] ;
  10:   ;
  11:  WAIT DI[12] = ON ;
  12:  IF DI[10] = ON,JMP LBL[2]  ;
  13:  LBL[10:Get clean tool] ;
  14:  IF R[11] <> 1 AND DI[73] =
OFF,JMP LBL[18]  ;
  15:   ;
  16:  LBL[2] ;
  17:  IF R[11] > R[15] AND DI[73] =
ON,JMP LBL[99]  ;
  18:  IF R[11] > R[15] AND DI[10] =
ON,JMP LBL[99]  ;
  19:  IF DI[73] = ON,JMP LBL[20]  ;
  20:  IF R[11] = 1 AND DI[10] =
ON,JMP LBL[200]  ;
  21:   ;
  22:  LBL[200] ;
  23:   ;
  24:  PR[13] =  UTOOL[3] ;
  25:   ;
  26:  CALL CLOS_CLN ;
  27:   ;
  28:   UTOOL[4] = PR[13] ;
  29:   ;
  30:  R[9] = 73 ;
  31:   ;
  32:  IF DI[50] = ON,JMP LBL[15]  ;
  33:  IF DI[51] = ON,JMP LBL[16]  ;
  34:  IF DI[10] = ON,JMP LBL[15]  ;
  35:  DO[22] = ON ;
  36:   ;
  37:  UALM[5] ;
  38:   ;
  39:  CALL MAINTEN ;
  40:  WAIT DI[53] = ON OR DI[54] =
ON OR DI[55] = ON ;
  41:  DO[22] = OFF ;
  42:  JMP LBL[14]  ;
  43:   ;
  44:  LBL[15:dry tool1] ;
  45:  PR[4] = P[31] ;
  46:  R[8] = 50 ;
  47:  JMP LBL[100]  ;
  48:  LBL[16:dry tool2] ;
  49:  PR[4] = P[32] ;
  50:  R[8] = 51 ;
  51:  JMP LBL[100]  ;
  52:   ;
  53:  LBL[18:Clean tool error] ;
  54:  DO[57] = ON ;
  55:  DO[15] = ON ;
  56:  WAIT DI[73] = ON ;
  57:  DO[15] = OFF ;
```

```
 58:  DO[57] = OFF ;
 59:  JMP LBL[10]  ;
 60:   ;
 61:  LBL[19:Reject tool] ;
 62:  R[14] = 1 ;
 63:  CALL REJECT ;
 64:  R[14] = 0 ;
 65:  JMP LBL[10]  ;
 66:   ;
 67:  LBL[20:Get wash tool] ;
 68:   ;
 69:  R[9] = 74 ;
 70:   ;
 71:  IF DI[R[9]] = ON,JMP LBL[110]  ;
 72:  CALL OPEN_CLN ;
 73:  CALL OPEN_GRP ;
 74:   ;
 75:  PR[13] =  UTOOL[2] ;
 76:  IF DI[57] = ON,JMP LBL[11]  ;
 77:  IF DI[58] = ON,JMP LBL[12]  ;
 78:  IF DI[10] = ON,JMP LBL[11]  ;
 79:  DO[22] = ON ;
 80:   ;
 81:  UALM[5] ;
 82:   ;
 83:  CALL MAINTEN ;
 84:  WAIT DI[57] = ON OR DI[58] = ON ;
 85:  DO[22] = OFF ;
 86:  JMP LBL[10]  ;
 87:   ;
 88:  LBL[11:Wash tool 1] ;
 89:  PR[4] = P[21] ;
 90:  R[8] = 57 ;
 91:  JMP LBL[100]  ;
 92:   ;
 93:  LBL[12:Wash tool 2] ;
 94:  PR[4] = P[22] ;
 95:  R[8] = 58 ;
 96:  JMP LBL[100]  ;
 97:   ;
 98:  LBL[100:Start] ;
 99:  PR[13,12] = PR[13,12] + 300 ;
100:   UTOOL[4] = PR[13] ;
101:L PR[4] 1500mm/sec FINE  ;
102:   ;
103:  PR[13,12] = PR[13,12] - 300 ;
104:   UTOOL[4] = PR[13] ;
105:L PR[4] 100mm/sec FINE  ;
106:  IF DI[10] = ON,JMP LBL[101]  ;
107:  WAIT DI[R[8]] = ON AND
DI[R[9]] = ON TIMEOUT,LBL[500] ;
108:   ;
109:  LBL[101:Sensor alarm ret] ;
110:   ;
111:  IF R[9] = 73,JMP LBL[105]  ;
112:   ;
113:  LBL[104] ;
114:   ;
115:  CALL CLOS_GRP ;
116:   ;
117:  LBL[105] ;
118:   ;
119:  PR[13,12] = PR[13,12] + 300 ;
120:   UTOOL[4] = PR[13] ;
121:L PR[4] 500mm/sec FINE  ;
122:   ;
123:  IF DI[10] = ON,JMP LBL[107]  ;
124:  WAIT DI[R[8]] = OFF AND
DI[R[9]] = ON TIMEOUT,LBL[500] ;
125:   ;
126:  LBL[107] ;
127:  IF DI[10] = ON,JMP LBL[99]  ;
128:  IF DI[73] = ON AND DI[74] =
ON,JMP LBL[99]  ;
129:  JMP LBL[20]  ;
130:   ;
131:  LBL[99:END] ;
132:  R[8] = 0 ;
133:  R[9] = 0 ;
134:  END ;
135:   ;
136:  LBL[500:Sensor alarm] ;
137:  DO[22] = ON ;
138:   ;
139:  UALM[4] ;
140:   ;
141:  CALL MAINTEN ;
142:  DO[22] = OFF ;
143:  JMP LBL[101]  ;
144:   ;
```

```
 145:  !Tool adquire points ;
 146:  !Wash tool point ;
 147:  UTOOL_NUM = 2 ;
 148:  UFRAME_NUM = 3 ;
 149:L P[21:Wash Tool1] 100mm/sec FINE
;
 150:L P[22:Wash Tool2] 100mm/sec FINE
;
 151:  !Dry tool point ;
 152:  UTOOL_NUM = 3 ;
 153:  UFRAME_NUM = 3 ;
 154:L P[31:Dry Tool1] 100mm/sec FINE  ;
 155:L P[32:Dry Tool2] 100mm/sec FINE  ;
```

8.2.10 GLUE_CYL

```
/PROG  GLUE_CYL MACRO
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Activate cylindr";
PROG_SIZE   = 378;
CREATE                = DATE 08-12-16
TIME 20:16:16;
MODIFIED    = DATE 08-12-16  TIME
20:16:16;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 16;
MEMORY_SIZE       = 786;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = *,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Activate posmo cylinder ;
   2:   ;
   3:   ;
   4:  LBL[10] ;
   5:  IF DO[42] = OFF,JMP LBL[20]  ;
   6:  DO[42] = OFF ;
   7:  IF DO[42] = OFF,JMP LBL[99]  ;
   8:   ;
   9:  LBL[20] ;
  10:  IF DO[42] = ON,JMP LBL[10]  ;
  11:  DO[42] = ON ;
  12:  IF DO[42] = ON,JMP LBL[99]  ;
  13:   ;
  14:  LBL[99] ;
  15:   ;
  16:  END ;
/POS
/END
```

8.2.11 HOME_POS

```
/PROG  HOME_POS MACRO
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Home position";
PROG_SIZE   = 549;
CREATE                = DATE 07-11-19
TIME 17:59:11;
MODIFIED    = DATE 08-12-12  TIME
11:57:00;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 10;
MEMORY_SIZE       = 849;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE      = 0,
     BUSY_LAMP_OFF       = 0,
     ABORT_REQUEST       = 0,
     PAUSE_REQUEST       = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/MN
   1:  !Home position ;
   2:   ;
   3:  UTOOL_NUM = 1 ;
   4:  PAYLOAD[1] ;
   5:  UFRAME_NUM = 1 ;
   6:   ;
   7:J P[1:Home position] 100% FINE  ;
   8:  DO[9] = ON ;
   9:  DO[14] = OFF ;
  10:  DO[16] = OFF ;
```

8.2.12 MAIN

```
/PROG  MAIN
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Main program";
PROG_SIZE   = 958;
CREATE                = DATE 06-08-04
TIME 12:47:13;
MODIFIED    = DATE 08-12-12  TIME
14:02:05;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 45;
MEMORY_SIZE       = 1434;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = *,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
  SPOT : TRUE ;
/MN
   1:  !Main program ;
   2:   ;
   3:  !Setting initial conditions ;
   4:  CALL INIT_CON ;
   5:   ;
   6:  LBL[1: Start sequence] ;
   7:   ;
   8:  !Robot home position check ;
   9:  CALL CHK_HOME ;
  10:   ;
  11:  IF DI[10] = ON,JMP LBL[5]  ;
  12:  WAIT DI[9] = ON ;
  13:   ;
  14:  !Setting cycle time timer ;
  15:  TIMER[1] = RESET ;
  16:  TIMER[1] = START ;
  17:   ;
  18:  !Reject tools routine ;
  19:   ;
  20:  LBL[5:Reject tools] ;
  21:   ;
  22:  IF DI[73] = ON,JMP LBL[11]  ;
  23:  IF DI[74] = ON,JMP LBL[11]  ;
  24:  JMP LBL[15]  ;
  25:   ;
  26:  LBL[11:Reject wash tool] ;
  27:  CALL REJECT ;
  28:  JMP LBL[5]  ;
  29:   ;
  30:  !Model selection routine ;
  31:  LBL[15] ;
  32:  CALL SELECT ;
  33:  CALL REJECT ;
  34:  !End of sequence control ;
  35:  CALL END_SEQ ;
  36:   ;
  37:  !Robot  to maintenance position ;
  38:  CALL MAINTEN ;
  39:   ;
  40:  !Robot home position  ;
  41:  CALL HOME_POS ;
  42:  DO[11] = ON ;
  43:  WAIT      3.00(sec) ;
  44:  DO[11] = OFF ;
  45:  JMP LBL[1]  ;
/POS
/END
```

8.2.13 MAINTEN

```
/PROG  MAINTEN
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Maintenance pos";
PROG_SIZE   = 709;
CREATE                = DATE 07-11-19
TIME 17:47:27;
MODIFIED    = DATE 08-11-13  TIME
18:17:19;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 19;
MEMORY_SIZE       = 1121;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/MN
   1:  !Maintenance position ;
   2:   ;
   3:  UTOOL_NUM = 1 ;
   4:  PAYLOAD[1] ;
   5:  UFRAME_NUM = 1 ;
   6:   ;
   7:  IF DI[14] = OFF,JMP LBL[99]  ;
   8:  CALL WAIT_POS ;
   9:  CALL HOME_POS ;
  10:   ;
  11:J P[1:Maintenance pos] 100% FINE  ;
  12:  DO[14] = ON ;
  13:  PAUSE ;
  14:  DO[14] = OFF ;
  15:   ;
  16:  !Robot home position  ;
  17:  CALL HOME_POS ;
  18:   ;
  19:  LBL[99:END] ;
```

8.2.14 NGET_WAS

```
/PROG  NGET_WAS
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Unget wash tool";
PROG_SIZE   = 2238;
CREATE                = DATE 08-11-11
TIME 18:26:07;
MODIFIED    = DATE 08-12-10  TIME
12:16:14;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 96;
MEMORY_SIZE       = 2734;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Get wash & dry tools from table ;
   2:  UTOOL_NUM = 4 ;
   3:  PAYLOAD[2] ;
   4:  UFRAME_NUM = 3 ;
   5:   ;
   6:  PR[11] =  UTOOL[1] ;
   7:  PR[12] =  UTOOL[2] ;
   8:  PR[14] =  UTOOL[3] ;
   9:   UTOOL[4] = PR[12] ;
  10:   ;
  11:  WAIT DI[12] = ON ;
  12:   ;
  13:  LBL[10:Get clean tool] ;
  14:  IF DI[74] = OFF,JMP LBL[99]  ;
  15:   ;
  16:  PR[13] =  UTOOL[3] ;
  17:  LBL[20:Unget wash tool] ;
  18:   ;
  19:  R[9] = 74 ;
  20:   ;
  21:  CALL OPEN_CLN ;
  22:   ;
  23:  PR[13] =  UTOOL[2] ;
  24:  IF DI[57] = OFF,JMP LBL[11]  ;
  25:  IF DI[58] = OFF,JMP LBL[12]  ;
  26:  DO[22] = ON ;
  27:   ;
  28:  UALM[5] ;
  29:   ;
  30:  CALL MAINTEN ;
  31:  WAIT DI[57] = OFF OR DI[58] =
OFF ;
  32:  DO[22] = OFF ;
  33:  JMP LBL[10]  ;
  34:   ;
  35:  LBL[11:Wash tool 1] ;
  36:  PR[4] = P[21] ;
  37:  R[8] = 57 ;
  38:  JMP LBL[100]  ;
  39:   ;
  40:  LBL[12:Wash tool 2] ;
  41:  PR[4] = P[22] ;
  42:  R[8] = 58 ;
  43:  JMP LBL[100]  ;
  44:   ;
  45:  LBL[100:Start] ;
  46:  PR[13,12] = PR[13,12] + 300 ;
  47:   UTOOL[4] = PR[13] ;
  48:L PR[4] 1500mm/sec FINE  ;
  49:   ;
  50:  PR[13,12] = PR[13,12] - 300 ;
  51:   UTOOL[4] = PR[13] ;
  52:L PR[4] 100mm/sec FINE  ;
  53:   ;
  54:  WAIT DI[R[8]] = ON AND
DI[R[9]] = ON TIMEOUT,LBL[500] ;
  55:   ;
  56:  LBL[104] ;
  57:   ;
  58:  CALL OPEN_GRP ;
  59:   ;
  60:  LBL[105] ;
  61:   ;
```

```
  62:  PR[13,12] = PR[13,12] + 300 ;
  63:   UTOOL[4] = PR[13] ;
  64:L PR[4] 500mm/sec FINE  ;
  65:   ;
  66:  WAIT DI[R[8]] = ON AND DI[R[9]]
= OFF TIMEOUT,LBL[500] ;
  67:   ;
  68:  LBL[107] ;
  69:  IF DI[73] = ON AND DI[74] =
OFF,JMP LBL[99]  ;
  70:  JMP LBL[20]  ;
  71:   ;
  72:  LBL[99:END] ;
  73:  R[8] = 0 ;
  74:  R[9] = 0 ;
  75:  END ;
  76:   ;
  77:  LBL[500:Sensor alarm] ;
  78:  DO[22] = ON ;
  79:   ;
  80:  UALM[4] ;
  81:   ;
  82:  CALL MAINTEN ;
  83:  DO[22] = OFF ;
  84:  JMP LBL[10]  ;
  85:   ;
  86:  !Tool adquire points ;
  87:  !Wash tool point ;
  88:  UTOOL_NUM = 2 ;
  89:  UFRAME_NUM = 3 ;
  90:L P[21:Wash Tool1] 100mm/sec FINE
;
  91:L P[22:Wash tool2] 100mm/sec FINE  ;
  92:  !Dry tool point ;
  93:  UTOOL_NUM = 3 ;
  94:  UFRAME_NUM = 3 ;
  95:L P[31:Dry Tool1] 100mm/sec FINE  ;
  96:L P[32:Dry Tool2] 100mm/sec FINE  ;
```

8.2.15 OPEN_CLN

```
/PROG  OPEN_CLN MACRO
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Open clean gripp";
PROG_SIZE  = 606;
CREATE                = DATE 08-03-12
TIME 16:15:12;
MODIFIED    = DATE 08-04-22  TIME
14:55:09;
FILE_NAME  = ;
VERSION     = 0;
LINE_COUNT          = 34;
MEMORY_SIZE       = 1070;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE      = 0,
     BUSY_LAMP_OFF       = 0,
     ABORT_REQUEST       = 0,
     PAUSE_REQUEST       = 0;
DEFAULT_GROUP  = *,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Open gripper ;
   2:   ;
   3:  DO[77] = OFF ;
   4:  DO[78] = ON ;
   5:   ;
   6:  LBL[1] ;
   7:  WAIT DI[76] = OFF AND DI[75] =
ON TIMEOUT,LBL[10] ;
   8:  DO[78] = OFF ;
   9:   ;
  10:  END ;
  11:   ;
  12:  LBL[10:Error handler] ;
  13:  DO[15] = ON ;
  14:   ;
  15:  IF DI[76] = OFF,JMP LBL[20]  ;
  16:   ;
  17:  DO[58] = ON ;
  18:  UALM[3] ;
  19:  WAIT DI[76] = OFF ;
  20:  DO[58] = OFF ;
  21:   ;
  22:  JMP LBL[1]  ;
  23:   ;
  24:  LBL[20] ;
  25:  IF DI[76] = ON,JMP LBL[99]  ;
  26:  DO[59] = ON ;
  27:  UALM[3] ;
  28:  WAIT DI[75] = ON ;
  29:  DO[59] = OFF ;
  30:   ;
  31:  LBL[99:END] ;
  32:  DO[15] = OFF ;
  33:   ;
  34:  JMP LBL[1]  ;
/POS
/END
```

8.2.16 OPEN_GRP

```
/PROG  OPEN_GRP MACRO
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Open gripper";
PROG_SIZE  = 594;
CREATE                = DATE 07-11-19
TIME 17:37:25;
MODIFIED    = DATE 08-07-28  TIME
14:54:10;
FILE_NAME = ;
VERSION     = 0;
LINE_COUNT            = 36;
MEMORY_SIZE           = 1050;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF          = 0,
     ABORT_REQUEST          = 0,
     PAUSE_REQUEST          = 0;
DEFAULT_GROUP  = *,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/MN
   1:  !Open gripper ;
   2:   ;
   3:  IF DI[68] = ON,JMP LBL[1]  ;
   4:   ;
   5:  DO[71] = OFF ;
   6:  DO[72] = ON ;
   7:   ;
   8:  LBL[1] ;
   9:  WAIT DI[67] = OFF AND DI[68] =
ON TIMEOUT,LBL[10] ;
  10:  DO[72] = OFF ;
  11:   ;
  12:  DO[72] = OFF ;
  13:   ;
  14:  END ;
  15:   ;
  16:  LBL[10:Error handler] ;
  17:  DO[15] = ON ;
  18:   ;
  19:  IF DI[68] = OFF,JMP LBL[20]  ;
  20:   ;
  21:  DO[63] = ON ;
  22:  UALM[6] ;
  23:  WAIT DI[67] = OFF ;
  24:  DO[63] = OFF ;
  25:   ;
  26:  LBL[20] ;
  27:  IF DI[68] = ON,JMP LBL[99]  ;
  28:  DO[63] = ON ;
  29:  UALM[6] ;
  30:  WAIT DI[68] = ON ;
  31:  DO[63] = OFF ;
  32:   ;
  33:  LBL[99:END] ;
  34:  DO[15] = OFF ;
  35:   ;
  36:  JMP LBL[1]  ;
/POS
/END
```

8.2.17 REJECT

```
/PROG  REJECT
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Reject bin pos";
PROG_SIZE  = 2540;
CREATE                = DATE 07-11-29
TIME 10:07:13;
MODIFIED    = DATE 08-12-11  TIME
13:01:01;
FILE_NAME  = ;
VERSION     = 0;
LINE_COUNT          = 126;
MEMORY_SIZE       = 2932;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE      = 0,
     BUSY_LAMP_OFF       = 0,
     ABORT_REQUEST       = 0,
     PAUSE_REQUEST       = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/MN
   1:  !Unload gripper in reject bin ;
   2:   ;
   3:  PAYLOAD[1] ;
   4:  UTOOL_NUM = 4 ;
   5:  UFRAME_NUM = 3 ;
   6:   ;
   7:  LBL[1] ;
   8:  IF DI[73] = ON,JMP LBL[10]  ;
   9:  IF DI[74] = ON,JMP LBL[20]  ;
  10:  IF DI[10] = ON,JMP LBL[10]  ;
  11:  IF DI[73] = OFF AND DI[74] =
OFF,JMP LBL[99]  ;
  12:   ;
  13:  LBL[10] ;
  14:  CALL CLOS_CLN ;
  15:  PR[4] = P[1] ;
  16:  PR[12] =  UTOOL[3] ;
  17:  PR[13] =  UTOOL[3] ;
  18:  PR[13,11] = PR[13,11] - 200 ;
  19:  PR[13,12] = PR[13,12] + 200 ;
  20:   UTOOL[4] = PR[13] ;
  21:  UTOOL_NUM = 4 ;
22:L PR[4] 500mm/sec FINE  ;
  23:  PR[13,12] = PR[13,12] - 200 ;
  24:   UTOOL[4] = PR[13] ;
  25:  UTOOL_NUM = 4 ;
26:L PR[4] 500mm/sec FINE  ;
  27:   ;
  28:  PR[13,11] = PR[13,11] + 200 ;
  29:   UTOOL[4] = PR[13] ;
  30:  UTOOL_NUM = 4 ;
31:L PR[4] 100mm/sec FINE  ;
  32:   ;
  33:  PR[13,12] = PR[13,12] + 200 ;
  34:   UTOOL[4] = PR[13] ;
  35:  UTOOL_NUM = 4 ;
36:L PR[4] 100mm/sec FINE  ;
  37:   ;
  38:  IF DI[10] = ON,JMP LBL[20]  ;
39: Undefined instruction;
  40:  IF DI[73] = OFF,JMP LBL[1]  ;
  41:   ;
  42:  !Part in gripper check ;
  43:   ;
  44:  DO[57] = ON ;
  45:  DO[15] = ON ;
  46:  ! ;
  47:  !Gripper with part ;
  48:  ! ;
  49:  !Unload part manually ;
  50:  ! ;
  51:  DO[22] = ON ;
  52:  CALL MAINTEN ;
  53:   ;
  54:  UALM[2] ;
  55:   ;
  56:  WAIT DI[73] = OFF ;
  57:  DO[15] = OFF ;
  58:  DO[57] = OFF ;
  59:  DO[22] = OFF ;
  60:  IF DI[74] = ON,JMP LBL[20]  ;
  61:   ;
  62:  JMP LBL[99]  ;
  63:   ;
```

```
64:  LBL[20:Reject wash tool] ;
65:  CALL OPEN_CLN ;
66:  PR[4] = P[2] ;
67:  PR[12] =  UTOOL[3] ;
68:  PR[13] =  UTOOL[3] ;
69:  PR[13,12] = PR[13,12] + 200 ;
70:   UTOOL[4] = PR[13] ;
71:  UTOOL_NUM = 4 ;
72:L PR[4] 500mm/sec FINE  ;
73:   ;
74:  PR[13,12] = PR[13,12] - 200 ;
75:   UTOOL[4] = PR[13] ;
76:  UTOOL_NUM = 4 ;
77:L PR[4] 100mm/sec FINE  ;
78:  CALL OPEN_GRP ;
79:  CALL ACT_CYL ;
80:  CALL ACT_CYL ;
81:   ;
82:  PR[13,12] = PR[13,12] + 200 ;
83:   UTOOL[4] = PR[13] ;
84:  UTOOL_NUM = 4 ;
85:L PR[4] 100mm/sec FINE  ;
86: Undefined instruction;
87:  IF DI[74] = ON,JMP LBL[21]  ;
88:   ;
89:  IF DI[73] = OFF,JMP LBL[99]  ;
90:  LBL[98] ;
91:  !Reset gripper conditions ;
92:  CALL OPEN_CLN ;
93:  CALL OPEN_GRP ;
94:  CALL ACT_CYL ;
95:   ;
96:  LBL[21] ;
97:  !Part in gripper check ;
98:   ;
99:  DO[64] = ON ;
100:  DO[15] = ON ;
101:  ! ;
102:  !Gripper with part ;
103:  ! ;
104:  !Unload part manually ;
105:  ! ;
106:  DO[22] = ON ;
107:  CALL MAINTEN ;
108:   ;
109:  UALM[2] ;
110:   ;
111:  WAIT DI[74] = OFF ;
112:  DO[15] = OFF ;
113:  DO[64] = OFF ;
114:  DO[22] = OFF ;
115:  IF DI[73] = ON,JMP LBL[10]  ;
116:   ;
117:  LBL[99:END] ;
118:   ;
119:  END ;
120:   ;
121:  !Reject point ;
122:  UTOOL_NUM = 3 ;
123:  UFRAME_NUM = 3 ;
124:   ;
125:L P[2:Reject position] 100mm/sec
FINE  ;
126:L P[1:] 100mm/sec FINE  ;
```

8.2.18 SELECT

```
/PROG  SELECT
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Model selection";
PROG_SIZE  = 3566;
CREATE                = DATE 07-11-21
TIME 17:18:20;
MODIFIED    = DATE 08-12-16  TIME
16:50:22;
FILE_NAME  = ;
VERSION     = 0;
LINE_COUNT          = 207;
MEMORY_SIZE       = 4262;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE      = 0,
     BUSY_LAMP_OFF       = 0,
     ABORT_REQUEST       = 0,
     PAUSE_REQUEST       = 0;
DEFAULT_GROUP  = *,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/MN
   1:  !Model selection routine ;
   2:   ;
   3:  !Getting model data ;
   4:  R[1] = GI[1] ;
   5:   ;
   6:  IF R[1] = 0,JMP LBL[99]  ;
   7:   ;
   8:  !Sequence conditions ;
   9:   ;
  10:  R[8] = 100 ;
  11:  R[9] = 100 ;
  12:  R[10] = 0 ;
  13:  R[11] = 1 ;
  14:  R[12] = 0 ;
  15:  R[13] = 0 ;
  16:  R[14] = 0 ;
  17:   ;
  18:  !Part offset position calculation ;
  19:  CALL OFF_CALC ;
  20:   ;
  21:  LBL[1000:Start sequence] ;
  22:   ;
  23:  !Model selection ;
  24:  SELECT R[1] = 1,JMP LBL[10]  ;
  25:         = 2,JMP LBL[20]  ;
  26:         = 3,JMP LBL[30]  ;
  27:         = 4,JMP LBL[40]  ;
  28:         = 5,JMP LBL[50]  ;
  29:         = 6,JMP LBL[60]  ;
  30:         = 7,JMP LBL[70]  ;
  31:         = 8,JMP LBL[80]  ;
  32:         = 9,JMP LBL[90]  ;
  33:         = 10,JMP LBL[100]  ;
  34:         = 11,JMP LBL[110]  ;
  35:         = 12,JMP LBL[120]  ;
  36:         = 13,JMP LBL[130]  ;
  37:         = 14,JMP LBL[140]  ;
  38:         = 15,JMP LBL[150]  ;
  39:  JMP LBL[99]  ;
  40:   ;
  41:  !********************** ;
  42:  !Model selection section ;
  43:   ;
  44:  LBL[10:GE left] ;
  45:  !Ref 721Z3700 ;
  46:  CALL RR1_LH ;
  47:  JMP LBL[98]  ;
  48:   ;
  49:  LBL[20:GE right] ;
  50:  !Ref 721Z3800 ;
  51:  CALL RR1_RH ;
  52:  JMP LBL[98]  ;
  53:   ;
  54:  LBL[30:RR left] ;
  55:  !Ref 725Z3700 ;
  56:  CALL GE1_LH ;
  57:  JMP LBL[98]  ;
  58:   ;
  59:  LBL[40:RR right] ;
  60:  !Ref 725Z3800 ;
  61:  CALL GE1_RH ;
  62:  JMP LBL[98]  ;
  63:   ;
  64:  LBL[50] ;
```

```
 65:  !Ref  ;
 66:   ;
 67:  JMP LBL[98]  ;
 68:   ;
 69:  LBL[60] ;
 70:  !Ref  ;
 71:   ;
 72:  JMP LBL[98]  ;
 73:   ;
 74:  LBL[70] ;
 75:  !Ref  ;
 76:   ;
 77:  JMP LBL[98]  ;
 78:   ;
 79:  LBL[80] ;
 80:  !Ref  ;
 81:   ;
 82:  JMP LBL[98]  ;
 83:   ;
 84:  LBL[90] ;
 85:  !Ref  ;
 86:   ;
 87:  JMP LBL[98]  ;
 88:   ;
 89:  LBL[100] ;
 90:  !Ref  ;
 91:   ;
 92:  JMP LBL[98]  ;
 93:   ;
 94:  LBL[110] ;
 95:  !Ref  ;
 96:   ;
 97:  JMP LBL[98]  ;
 98:   ;
 99:  LBL[120] ;
100:  !Ref  ;
101:   ;
102:  JMP LBL[98]  ;
103:   ;
104:  LBL[130] ;
105:  !Ref  ;
106:   ;
107:  JMP LBL[98]  ;
108:   ;
109:  LBL[140] ;
110:  !Ref  ;
111:   ;
112:  JMP LBL[98]  ;
113:   ;
114:  LBL[150] ;
115:  !Ref DUMMY ;
116:  CALL GE1_LH ;
117:  JMP LBL[98]  ;
118:   ;
119:   ;
120:  !********************** ;
121:  LBL[98:Change work zone] ;
122:  IF R[11] = R[16] OR R[11] =
R[17],JMP LBL[200]  ;
123:   ;
124:  LBL[170:Clean routine] ;
125:   ;
126:  IF R[13] >= 1,JMP LBL[180]  ;
127:   ;
128:  !Clean all Click_bond positions ;
129:  CALL CLOS_CLN ;
130:  IF DI[73] = ON AND DI[74] =
ON,JMP LBL[171]  ;
131:  IF DI[10] = ON AND R[11] <>
1,JMP LBL[171]  ;
132:  CALL GET_ABRA ;
133:  CALL WAIT_POS ;
134:   ;
135:  LBL[171] ;
136:  CALL ABRA_ROU ;
137:  R[11] = R[11] + 1 ;
138:  IF R[11] <= R[15],JMP LBL[1000]
;
139:  CALL WAIT_POS ;
140:  R[14] = 1 ;
141:  CALL REJECT ;
142:  R[11] = 1 ;
143:  R[13] = 1 ;
144:  JMP LBL[1000]  ;
145:   ;
146:  !Get tools & clean 3 click bonds ;
147:  LBL[180:Wash & dry rout] ;
148:  IF R[13] = 2,JMP LBL[198]  ;
149:   ;
150:  LBL[188:Clean sequence] ;
```

```
151:  !Clean 3 clickbond positions ;
152:  IF R[12] > 0,JMP LBL[189]  ;
153:  CALL GET_WASH ;
154:  CALL WASH_APP ;
155:  IF R[12] <> 0,JMP LBL[189]  ;
156:  CALL OUT_POIN ;
157:  CALL WAIT_POS ;
158:   ;
159:  LBL[189] ;
160:  CALL CLN_ROUT ;
161:  R[11] = R[11] + 1 ;
162:  R[12] = R[12] + 1 ;
163:   ;
164:   ;
165:  !End sequence conditions ;
166:  IF R[11] > R[15],JMP LBL[197]  ;
167:   ;
168:  IF R[12] < 10,JMP LBL[1000]  ;
169:   ;
170:   ;
171:  LBL[197] ;
172:  !Init conditions ;
173:  R[11] = R[11] - R[12] ;
174:  R[13] = 2 ;
175:  CALL WAIT_POS ;
176:  CALL NGET_WAS ;
177:  JMP LBL[1000]  ;
178:   ;
179:  LBL[198:Fix sequence] ;
180:  !Pick click bond from table ;
181:  CALL DEPALLET ;
182:   ;
183:  !Bond application ;
184:  CALL BOND_APP ;
185:  CALL OUT_POIN ;
186:  CALL WAIT_POS ;
187:   ;
188:  !Fix click bond on part ;
189:  CALL FIX_ROUT ;
190:  CALL WAIT_POS ;
191:  R[11] = R[11] + 1 ;
192:  R[12] = R[12] - 1 ;
193:   ;
194:  !End sequence conditions ;
195:  IF R[11] > R[15],JMP LBL[99]  ;
196:  IF R[12] > 0,JMP LBL[1000]  ;
197:  R[13] = 1 ;
198:  JMP LBL[1000]  ;
199:   ;
200:  LBL[200] ;
201:  CALL WAIT_POS ;
202:  JMP LBL[170]  ;
203:   ;
204:  LBL[99:END] ;
205:  IF DI[74] = OFF,CALL
NGET_WAS ;
206:  !Robot home position  ;
207:  CALL HOME_POS ;
/POS
/END
```

8.2.19 WAIT_POS

```
/PROG  WAIT_POS
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Wait pos point";
PROG_SIZE   = 582;
CREATE                = DATE 08-02-21
TIME 12:14:28;
MODIFIED    = DATE 08-12-12  TIME
12:07:19;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 7;
MEMORY_SIZE       = 894;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/MN
   1:  !Go to wait position ;
   2:   ;
   3:  UTOOL_NUM = 1 ;
   4:  PAYLOAD[1] ;
   5:  UFRAME_NUM = 2 ;
   6:L P[1:Wait position] 1500mm/sec
CNT50  ;
   7:J P[2:Wait position ] 100% CNT50  ;
```

8.2.20 WASH_APP

```
/PROG  WASH_APP
/ATTR
OWNER                 = MNEDITOR;
COMMENT               = "Bonding station";
PROG_SIZE   = 866;
CREATE                = DATE 08-10-29
TIME 11:47:26;
MODIFIED    = DATE 08-12-12  TIME
12:12:20;
FILE_NAME  = ;
VERSION      = 0;
LINE_COUNT          = 18;
MEMORY_SIZE       = 1274;
PROTECT               = READ_WRITE;
TCD:  STACK_SIZE = 0,
     TASK_PRIORITY          = 50,
     TIME_SLICE       = 0,
     BUSY_LAMP_OFF        = 0,
     ABORT_REQUEST        = 0,
     PAUSE_REQUEST        = 0;
DEFAULT_GROUP  = 1,*,*,*,*;
CONTROL_CODE    = 00000000
00000000;
/APPL
/MN
   1:  !Click bond application ;
   2:   ;
   3:  UTOOL_NUM = 2 ;
   4:  PAYLOAD[1] ;
   5:  UFRAME_NUM = 3 ;
   6:   ;
   7:  LBL[100] ;
   8:   ;
   9:  CALL OPEN_CLN ;
  10:   ;
  11:J P[1:Aproching pos] 25% CNT100  ;
  12:   ;
  13:L P[3:Bond service pos] 300mm/sec
FINE  ;
  14:  WAIT      1.50(sec) ;
  15:L P[2:] 100mm/sec FINE  ;
  16:  WAIT      1.00(sec) ;
  17:L P[1:Aproching pos] 50mm/sec
CNT100  ;
  18:J P[4:Output point] 100% CNT100  ;
```

8.2 Anexo 3

8.3.1 Programa PLC

SIMATIC 300(2)\PLC 317-2DP\...\FC119 -<offline>

FC119 -<offline>

"RobotFanuc" Rotot Fanuc (Click bond)
Nombre: Familia:
Autor: Versión: 0.1
Versión del bloque: 2

Hora y fecha Código: 06/03/2009 12:21:30
Interface: 19/12/2007 17:12:46

Longitud (bloque / código / datos): 02188 01936 00004

Propiedades del objeto:

S7_language 9(1) Inglés (Estados Unidos)
22.05.2008 08:51:43

Nombre Tipo de datos Dirección
Comentario
IN 0.0
OUT 0.0
IN_OUT 0.0
TEMP 0.0
AUX_RESET_FALLOS Bool 0.0
AUX_PULSO_RESET Bool 0.1
ManEnable Bool 0.2
AutoEnable Bool 0.3
Reserve Bool 0.4
AbrasiveToolsOK Bool 0.5
CleaningToolsOK Bool 0.6
AcetoneToolsOK Bool 0.7
DryingToolsOK Bool 1.0
RETURN 0.0
RET_VAL 0.0

Bloque: FC119 ROBOT FANUC

Segm.: 1 Load Robot Inputs

U "Bit1"
= L 2.0
U L 2.0
SPBNB _001
L PEW 70
T DB360.DBW 16

_001: NOP 0
U L 2.0
SPBNB _002
L PEW 72
T DB360.DBW 18

_002: NOP 0
U L 2.0
SPBNB _003
L PEW 74
T DB360.DBW 20

_003: NOP 0
U L 2.0
SPBNB _004
L PEW 76
T DB360.DBW 22

_004: NOP 0

= "IORobot".HOLD

Segm.: 2 ROBOT FANUC

Segm.: 3 Manual mode Enable

U(
O "IORobot".T1_MODE
O "IORobot".T2_MODE
)
U "SafetyZ3OK"
= #ManEnable
Segm.: 4 Automatic mode Enable
U "IORobot".AUTO_MODE
U "SafetyZ3Started"
= #AutoEnable
Segm.: 5 ROBOT FANUC

Segm.: 6 Robot Fanuc Inmediate Stop
U "EmergOK"
U(
O #AutoEnable
O #ManEnable
)
UN "IORobot".COLLISION_STOP
= "IORobot".IMSTP
Segm.: 7 Robot Fanuc Safety speed
U "EmergOK"
U(
O #AutoEnable
O #ManEnable
)
UN "IORobot".COLLISION_STOP
= "IORobot".SFSPD
Segm.: 8 Robot Fanuc Aux. Stop program
U "IORobot".AUTO_MODE
UN "Ctrl-E".DEV_80.ACTIONS.Start_Pause
UN "IORobot".IN_SEQ
O "IORobot".COLLISION_STOP
= "STOP_ROBOT"

Segm.: 9 Robot Fanuc Hold input

UN "STOP_ROBOT"
U(
O #AutoEnable
O #ManEnable
)

```
Segm.: 10 Robot Fanuc Aux. Reset faults
U(
O "IORobot".AUTO_MODE
O "AUX_RESET_FAULTS"
)
U "SafetyZ3OK"
UN "IORobot".HELD
U(
ON "IORobot".SYSRDY
O
U "IORobot".FAULT
U "Ctrl-E".DEV_80.ACTIONS.Start_Pause
)
UN "IORobot".TPENABLE
= "AUX_RESET_FAULTS"
Segm.: 11 Robot Fanuc Reset faults
U "AUX_RESET_FAULTS"
U "Flashing"
= "IORobot".FAULT_RESET
Segm.: 12 Robot Fanuc Cycle stop
U(
U "IORobot".AUTO_MODE
U #AutoEnable
O #ManEnable
)
U "IORobot".CMDENBL
R "IORobot".CYCLE_STOP
Segm.: 13 PLC Bit Start
U "IORobot".AUTO_MODE
U "UST".PartOK
U "SH2".U.PosA
U "SH2".V.PosR
UN "RobEndOfJob"
= "PLC_START_ROBOT"

Segm.: 14 Robot Fanuc End of Job

U(
U "IORobot".AUTO_MODE
U "IORobot".END_OF_CYCLE
O
U "I_MCP_KeySwitch_3"
U "Ctrl-E".DEV_00.ACTIONS.Actions_11
)
S "RobEndOfJob"
U(
O "UST".Empty
O
U "I_MCP_KeySwitch_3"
U "Ctrl-E".DEV_00.ACTIONS.Actions_12
O
U "IORobot".SPARE_DO30
U "DRY_RUN"
)
R "RobEndOfJob"
NOP 0
```

```
Segm.: 15 Robot Fanuc End of Job

U "RobEndOfJob"
U "IORobot".END_OF_CYCLE
= "IORobot".ECO_END_OF_CYCLE
Segm.: 16 Aux.Start Program Robot Fanuc
U(
O "IORobot".HELD
O "IORobot".MAINT_POS
O "IORobot".IN_TOOL_CHG_POS
)
FP "FLSTART_ROBOT"
S "START_ROBOT_FANUC"
U(
U(
O "IORobot".TPENABLE
O "PC_Cycle_Start"
)
UN "IORobot".MAINT_POS
UN "IORobot".IN_TOOL_CHG_POS
O "SafetyZ3Started"
)
R "START_ROBOT_FANUC"
NOP 0
Segm.: 17 Robot Fanuc Start
U "IORobot".AUTO_MODE
U "IORobot".CMDENBL
U "IORobot".SYSRDY
UN "IORobot".PROGRUN
U "IORobot".HOLD
UN "START_ROBOT_FANUC"
L S5T#500MS
SI "TStartFanuc"
NOP 0
NOP 0
NOP 0
U "TStartFanuc"
= "IORobot".START
Segm.: 18 Robot Fanuc Enabled
O #AutoEnable
O #ManEnable
= "IORobot".ENABLE
Segm.: 19 Robot Fanuc Programm Start
O "PLC_START_ROBOT"
O
UN "IORobot".AUTO_MODE
U "DRY_RUN"
O
U "IORobot".AUTO_MODE
U(
U "ROBOT_MAINT"
UN "IORobot".MAINT_POS
O
U "ROBOT_TOOL_CHG"
UN "IORobot".IN_TOOL_CHG_POS
)
= "ROBOT_ST_PROGRAM"
```

```
Segm.: 20 Robot Fanuc Programm Start
U "ROBOT_ST_PROGRAM"
U "IORobot".CMDENBL
U "IORobot".PROGRUN
U "IORobot".ATPERCH
U "Ctrl-E".DEV_80.ACTIONS.Start_Pause
FP "RobAuxStartPulse"
= "IORobot".CYCLE_START
Segm.: 21 Robot Fanuc Dry Run
UN "IORobot".IN_SEQ
U "Ctrl-E".DEV_80.ACTIONS.Actions_02
U "I_MCP_KeySwitch_3"
S "DRY_RUN"
U "Ctrl-E".DEV_80.ACTIONS.Actions_03
R "DRY_RUN"
NOP 0
Segm.: 22 Robot Fanuc Dry Run
U "DRY_RUN"
= "IORobot".DRY_RUN
Segm.: 23 Robot Fanuc maintenance
position
U "IORobot".MAINTENANCE_REQUEST
UN "IORobot".MAINT_POS
U "Ctrl-E".DEV_80.ACTIONS.Actions_04
= "ROBOT_MAINT"
Segm.: 24 Robot Fanuc maintenance
position
O "IORobot".MAINT_POS
ON "IORobot".MAINTENANCE_REQUEST
R "Ctrl-E".DEV_80.ACTIONS.Actions_04
Segm.: 25 Robot Fanuc maintenance
position
U "ROBOT_MAINT"
= "IORobot".MAINT
Segm.: 26 Model bit 1
O "UST".M_RR_L_1
O "UST".M_GE_L_1
O "UST".M_Dummy
= "IORobot".MODELO_BIT1
Segm.: 27 Model bit 2
O "UST".M_RR_R_1
O "UST".M_GE_L_1
O "UST".M_Dummy
= "IORobot".MODELO_BIT2
```

```
Segm.: 28 Model bit 3
O "UST".M_GE_R_1
O "UST".M_Dummy
= "IORobot".MODELO_BIT3
Segm.: 29 Model bit 4
U "UST".M_Dummy
= "IORobot".MODELO_BIT4
Segm.: 30 Click table OK
U "I_TBR_S1AA"
U "I_TBR_S1AR"
U "I_TBR_S2AA"
U "I_TBR_S3AA"
U "I_TBR_S4AA"
U "I_TBR_S5AA"
= "IORobot".CLK_TABLE_OK
Segm.: 31 Tools presence
U "Bit1"
= L 2.0
U L 2.0
U "I_TBR_SPP7"
= "IORobot".CLEAN_TOOL3
U L 2.0
U "I_TBR_SPP8"
= "IORobot".CLEAN_TOOL2
U L 2.0
U "I_TBR_SPP10"
= "IORobot".CLEAN_TOOL1
U L 2.0
U "I_TBR_SPP12"
= "IORobot".DRY_TOOL3
U L 2.0
U "I_TBR_SPP13"
= "IORobot".DRY_TOOL2
U L 2.0
U "I_TBR_SPP14"
= "IORobot".DRY_TOOL1
U L 2.0
U "I_TBR_SPP9"
= "IORobot".TOOL3
U L 2.0
U "I_TBR_SPP11"
= "IORobot".TOOL2
U L 2.0
U "I_TBR_SPP15"
= "IORobot".TOOL1
Segm.: 32 Reset Click table OK
U "IORobot".ECHO_RST_CLK_TABLE
= "IORobot".RST_CLK_TABLE
Segm.: 33 Abrasive tools presence
U "Bit1"
U(
O "IORobot".TOOL1
O "IORobot".TOOL2
O "IORobot".TOOL3
)
= #AbrasiveToolsOK
```

= "Ctrl-E".DEV_80.STATUS.Robot_OK

Segm.: 34 Cleaning tools presence
U "Bit1"
U(
O "IORobot".CLEAN_TOOL1
O "IORobot".CLEAN_TOOL2
O "IORobot".CLEAN_TOOL3
)
= #CleaningToolsOK
Segm.: 35 Acetone tools presence
U "Bit1"
U(
O "IORobot".TOOL1
O "IORobot".TOOL2
O "IORobot".TOOL3
)
= #AcetoneToolsOK
Segm.: 36 Drying tools presence
U "Bit1"
U(
O "IORobot".DRY_TOOL1
O "IORobot".DRY_TOOL2
O "IORobot".DRY_TOOL3
)
= #DryingToolsOK
Segm.: 37 Activate Bond
U "IORobot".ACT_BOND
= "BOND_ACT"
Segm.: 38 Replace doser
U "IORobot".RPL_DOSER
= "DOSER_REP"
Segm.: 39 Ctrl-E

Segm.: 40 Ctrl-E Fanuc Automatic Mode
(1=Automatic, 0=Manual)
U "IORobot".AUTO_MODE
= "Ctrl-E".DEV_80.STATUS.Robot_mode
Segm.: 41 Ctrl-E Robot_Ready
U(
O #AutoEnable
O #ManEnable
)
U "IORobot".CLK_TABLE_OK
UN "IORobot".END_CLK_BOND
U #AbrasiveToolsOK
U #CleaningToolsOK
U #AcetoneToolsOK
U #DryingToolsOK

= "Ctrl-
E".DEV_80.STATUS.Fault_gripper

Segm.: 42 Ctrl-E Click bond table clamps
OK
U "IORobot".CLK_TABLE_OK
= "Ctrl-E".DEV_84.STATUS.Table_OK
Segm.: 43 Ctrl-E Click bonds in tABLE
UN "IORobot".END_CLK_BOND
= "Ctrl-E".DEV_84.STATUS.Clickbonds_OK
Segm.: 44 Ctrl-E Posmo OK
UN
"DB_EASYMASK".MASK131_POSMO.LED
S.error
= "Ctrl-E".DEV_86.STATUS.Posmo_OK
Segm.: 45 Ctrl-E Safety OK
O #AutoEnable
O #ManEnable
= "Ctrl-E".DEV_80.STATUS.Security_OK
Segm.: 46 Ctrl-E tools presence OK
U "Bit1"
= L 2.0
U L 2.0
U #AbrasiveToolsOK
= "Ctrl-
E".DEV_84.STATUS.Abrasive_tool_OK
U L 2.0
U #CleaningToolsOK
= "Ctrl-E".DEV_84.STATUS.Clean_tool_OK
U L 2.0
U #AcetoneToolsOK
= "Ctrl-E".DEV_84.STATUS.Wash_tool_OK
U L 2.0
U #DryingToolsOK
= "Ctrl-E".DEV_84.STATUS.Dry_tool_OK
Segm.: 47 Ctrl-E Program started
U "PLC_START_ROBOT"
S "Ctrl-E".DEV_80.STATUS.Job_Started
Segm.: 48 Ctrl-E Program finished
U "IORobot".END_OF_CYCLE
S "Ctrl-E".DEV_80.STATUS.Job_finished
Segm.: 49 Ctrl-E Gripper error
U "Bit1"
U(
O "IORobot".GRP_CLOSE_ERR
O "IORobot".GRP_OPEN_ERR
O "IORobot".CYL_FWD_ERR
O "IORobot".CYL_BWD_ERR
O "IORobot".SPP_ERR
)

SIMATIC P55023_IFS_A1\ 22/06/2009 16:29:38
SIMATIC 300(2)\PLC 317-2DP\...\FC119 - <offline>

Segm.: 50 Ctrl-E Clean clamp error

```
U "Bit1"
U(
O "IORobot".CLEAN_CLOSE_ERROR
O "IORobot".CLEAN_OPEN_ERROR
)
= "Ctrl-
E".DEV_80.STATUS.Fault_Clean_griper
```

Segm.: 51 Ctrl-E Robot Fanuc Dry Run

```
U "DRY_RUN"
= "Ctrl-E".DEV_80.STATUS.Status_06
```

Segm.: 52 Ctrl-E Robot Fanuc Dry Run

```
U "IORobot".MAINT_POS
= "Ctrl-E".DEV_80.STATUS.Status_07
```

Segm.: 53 Ctrl-E End of Job Robot Fanuc

```
U "RobEndOfJob"
= "Ctrl-E".DEV_00.STATUS.Status_21
```

Segm.: 54 Ctrl-E Start/Stop Robot

```
ON "I_A1_Z3_KG2"
ON "SafetyZ3OK"
O "UST".Empty
O
U "Ctrl-E".DEV_80.STATUS.Job_finished
UN "IORobot".SPARE_DO30
R "Ctrl-E".DEV_80.ACTIONS.Start_Pause
```

Segm.: 55 Load Robot Outputs

```
U "Bit1"
= L 2.0
U L 2.0
SPBNB _005
L DB360.DBW 0
T PAW 70

_005: NOP 0
U L 2.0
SPBNB _006
L DB360.DBW 2
T PAW 72

_006: NOP 0
U L 2.0
SPBNB _007
L DB360.DBW 4
T PAW 74

_007: NOP 0
U L 2.0
SPBNB _008
L DB360.DBW 6
T PAW 76

_008: NOP 0
```

Segm.: 56 Robot Alarms

SIMATIC P55023_IFS_A1\ 22/06/2009 16:29:38
SIMATIC 300(2)\PLC 317-2DP\...\FC119 - <offline>

Segm.: 57 ROB_S3AR Sensor Fault. Robot Fanuc clean clamp (DI76)
U "IORobot".CLEAN_CLOSE_ERROR
= "ALMSG_DB".A7004xx[12]
Segm.: 58 ROB_S3AA Sensor Fault. Robot Fanuc clean clamp (DI75)
U "IORobot".CLEAN_OPEN_ERROR
= "ALMSG_DB".A7004xx[13]
Segm.: 59 ROB_S1AA Sensor Fault. Robot Fanuc cylinder (DI65)
U "IORobot".CYL_FWD_ERR
= "ALMSG_DB".A7004xx[14]
Segm.: 60 ROB_S1AR Sensor Fault. Robot Fanuc cylinder (DI66)
U "IORobot".CYL_BWD_ERR
= "ALMSG_DB".A7004xx[15]
Segm.: 61 ROB_S2AA Sensor Fault. Robot Fanuc gripper (DI67)
U "IORobot".GRP_CLOSE_ERR
= "ALMSG_DB".A7004xx[16]
Segm.: 62 ROB_S2AR Sensor Fault. Robot Fanuc gripper (DI68)
U "IORobot".GRP_OPEN_ERR
= "ALMSG_DB".A7004xx[17]
Segm.: 63 ROB_SPP1 Sensor Fault. Robot Fanuc Click Bond (DI74)
U "IORobot".SPP_ERR
= "ALMSG_DB".A7004xx[18]
Segm.: 64 Info: Robot fanuc is not in safe position
U(
U(
O "SH2".U.ReqA
O "SH2".U.ReqR
)
U "SH2".V.PosA
O
U(
O "SH2".V.ReqA
O "SH2".V.ReqR
)
U "SH2".U.PosR
)
UN "IORobot".RELEASE_SHUTTLE
= "ALMSG_DB".A7004xx[50]