

AUTOMATIC TEST CREATION ENVIRONMENT WITH AN INDUSTRIAL ROBOT

Tommi Valkeapää

Bachelor's Thesis

May 2011

Degree Programme in Automation Engineering
Technology and Transport



JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES



Tekijä(t) VALKEAPÄÄ, Tommi	Julkaisun laji Opinnäytetyö	Päivämäärä 06.05.2011
	Sivumäärä 69	Julkaisun kieli Englanti
	Luottamuksellisuus () saakka	Verkkojulkaisulupa myönnetty (X)
Työn nimi AUTOMATIC TEST CREATION ENVIRONMENT WITH INDUSTRIAL ROBOT		
Koulutusohjelma Automaatiotekniikan koulutusohjelma		
Työn ohjaaja(t) RANTAPUSKA, Seppo		
Toimeksiantaja(t) EADS Secure Networks Oy MÄENPÄÄ, Ismo		
Tiivistelmä <p>Opinnäytetyö on raportti käytännön työstä, jossa kehitettiin EADS Secure Networks Oy:lle automaattinen testien luonti- ja ajoympäristö. EADS Secure Networks Oy kehittää TETRA-verkkoa käyttäviä päätelaitteita viranomaisten käyttöön. Opinnäytetyössä luodulla järjestelmällä testataan tilaajayrityksen tuotteiden osien mekaanista kestävyyttä.</p> <p>Opinnäytetyön tavoitteena oli kehittää mekaanisten testien luontiympäristö. Ympäristön on oltava helpokäyttöinen ja helposti laajennettava. Tilaajayritys kehittää jatkuvasti uusia tuotteita. Tämän takia järjestelmän on oltava laajennettavissa ja sovellettava myös vasta suunnitteilla oleville tuotteille.</p> <p>Opinnäytetyö esittelee järjestelmän laite- ja ohjelmisto-osat. Opinnäytetyö keskittyy tietokone- ja robottiohjelmistojen toteutuksen kuvaamiseen. Työssä esitellään FANUC Roboticsin PC Developers Kit-ohjelmistopakettien käyttöä LabVIEW-ohjelmiston kanssa. Opinnäytetyössä näiden ohjelmien avulla on luotu ympäristö, jolla robotin liikeohjelmia suoritetaan tietokoneen muistissa robottiohjaimen muistin sijaan.</p> <p>Lopullisen järjestelmän pääosat ovat FANUC Roboticsin LR MATE 200i -robotti ja R-J3-robottiohjain. Testien luontiin, robotin tilan valvontaan, liikepisteiden opetukseen ja projektitiedostojen hallintaan käytetään National Instrumentsin LabVIEW 8.5 ohjelmistoa.</p> <p>Tuloksia tarkastellessa voidaan todeta, että työn tavoitteet on saavutettu. Opinnäytetyön tuloksena on käyttövalmis testien luonti- ja ajoympäristö. Ympäristöllä on helppo luoda robottiohjelmia graafisen ohjelmointikielen ansiosta. Robotin tilan valvonta ja liikepisteiden opetus on helppoa luotujen sovellusten ansiosta. Järjestelmällä on suoritettu testejä, joista tilaajayritys on saanut tuotekehityksessä hyödynnettävää tietoa.</p>		
Avainsanat (asiasanat) ohjelmistosuunnittelu, testaus, automaatio, robotti, FANUC, LabVIEW, PCDK		
Muut tiedot		



Author(s) VALKEAPÄÄ, Tommi	Type of publication Bachelor's Thesis	Date 06.05.2011
	Pages 69	Language English
	Confidential <input type="checkbox"/> Until	Permission for web publication <input checked="" type="checkbox"/>
Title AUTOMATIC TEST CREATION ENVIRONMENT WITH INDUSTRIAL ROBOT		
Degree Programme Degree Programme in Automation Engineering		
Tutor(s) RANTAPUSKA, Seppo		
Assigned by EADS Secure Networks MÄENPÄÄ, Ismo		
Abstract <p>Thesis reports the actual project carried out when creating a mechanical test creation environment for EADS Secure Networks. EADS Secure Networks develops professional mobile radios that operate in TETRA Network. With the created system EADS Secure Networks tests their products mechanical endurance and wear resistance.</p> <p>The goal of the thesis was to create mechanical test creation environment. The environment must be easy to use and adaptable. EADS Secure networks continuously develops new products. The test creation environment must be suitable also for products in development phase.</p> <p>Thesis introduces the software and the hardware components of the system. The emphasis of this thesis is on host computer and robot software. It is described here how FANUC Robotics PC Developers Kit is used with LabVIEW. Additionally this thesis introduces how to create environment with LabVIEW that executes robot programs inside the computer's memory instead of the robot controller's memory.</p> <p>The main components of the system are FANUC Robotics LR Mate 200i industrial robot and R-J3 robot controller. Test creation, robot status monitoring, position teaching and project document management are handled with National Instruments LabVIEW 8.5.</p> <p>As for the results of this thesis, it can be stated that the goals were achieved. As a result of this thesis mechanical test development and execution environment was created. With the system it is easy to create new robot programs with graphical programming language. Robot monitoring and position teaching tasks are easy due to additional programs created. EADS Secure Networks have performed mechanical tests with the system and has gained useful information from the test results.</p>		
Keywords software design, testing, automation, robot, FANUC, LabVIEW, PCDK		
Miscellaneous		

CONTENTS

1	INTRODUCTION.....	6
1.1	GENERAL.....	6
1.2	GOALS.....	8
1.3	TOPICALITY AND IMPORTANCE	9
1.4	COMPANY DETAILS.....	10
2	THEORETICAL BACKGROUND.....	12
2.1	ROBOT PROGRAMMING.....	12
2.2	LABVIEW PROGRAMMING	13
2.3	ACTIVE X.....	15
2.4	MATS SOFTWARE COMPONENTS	15
2.4.1	<i>General.....</i>	15
2.4.2	<i>FANUC Robotics PC Developers kit.....</i>	17
2.5	ROBOT SERVER	18
2.5.1	<i>LabVIEW in Mechanics Automated Test System.....</i>	18
2.5.2	<i>Data communications option</i>	19
2.5.3	<i>Robot controller alarms</i>	20
2.6	MATS HARDWARE COMPONENTS	20
2.6.1	<i>General.....</i>	20
2.6.2	<i>FANUC Robotics LR Mate 200i industrial robot</i>	21
2.6.3	<i>FANUC Robotics R-J3 Robot controller</i>	22
2.6.4	<i>FANUC input and output modules</i>	23
2.6.5	<i>SMC Auto Hand Change System</i>	23
2.6.6	<i>LabJack U6 data acquisition card</i>	24
2.6.7	<i>HBM AE101 50N force transducer</i>	25
2.6.8	<i>SMC ZSE40 pressure switch</i>	25
3	MECHANICS AUTOMATED TEST SYSTEM	26
3.1	PROGRAM ARCHITECTURE.....	26
3.1.1	<i>Initializing and connecting the robot object.....</i>	26
3.1.2	<i>Reference number.....</i>	27
3.1.3	<i>Program execution order</i>	28
3.2	SUB PROGRAMS	30
3.2.1	<i>General.....</i>	30
3.2.2	<i>Move to example subVI.....</i>	30

3.3	LABVIEW SUBVI ICONS	35
3.4	POSITION TEACHING	36
3.4.1	<i>General</i>	36
3.4.2	<i>Teach positions LabVIEW program</i>	37
3.4.3	<i>Position format</i>	38
3.5	DATA COLLECTION	39
3.6	USER PANEL	39
3.7	TOOL IDENTIFICATION	42
3.8	TEST PROJECT HANDLING	43
3.9	ETHERNET SECURITY	45
4	EXAMPLE CASE.....	46
4.1	GENERAL.....	46
4.2	TEST DESIGN	46
4.3	DEFINITION	49
4.4	TEACH POSITIONS	51
4.5	PROGRAM CREATION	52
5	RESULTS	54
6	SYSTEMS FUNCTIONALITY AND USABILITY	55
7	REQUIREMENTS	56
8	CONCLUSIONS	58
8.1	GENERAL.....	58
8.2	FUTURE IMPROVEMENTS	58
8.3	ACHIEVED GOALS.....	59
9	SOURCES	60
10	APPENDICES.....	61
	APPENDIX 1. MATS BLOCK DIAGRAM.....	61
	APPENDIX 2. LR MATE 200I DATA SHEET.....	62
	APPENDIX 3. R-J3 DATA SHEET	64
	APPENDIX 4. EXAMPLE TEST PROGRAM.....	66
	APPENDIX 5. MATS USABILITY STUDY	67
	APPENDIX 6. TEACH POSITIONS LABVIEW PROGRAM.....	68
	APPENDIX 7. MATS USER PANEL	69

FIGURES

FIGURE 1. MATS BLOCK DIAGRAM	7
FIGURE 2. CASSIDIAN ORGANIZATION CHART.....	11
FIGURE 3. EXAMPLE OF DATA FLOW PROGRAMMING.....	13
FIGURE 4. LABVIEW OBJECTS.....	14
FIGURE 5. EXAMPLE FROM MATS BLOCK DIAGRAM.....	15
FIGURE 6. BLOCK DIAGRAM OF MATS SOFTWARE COMPONENTS.....	17
FIGURE 7. FANUC LR MATE 200i INDUSTRIAL ROBOT.....	21
FIGURE 8. FANUC ROBOTICS R-J3 ROBOT CONTROLLER.....	22
FIGURE 9. LABJACKU6 AND ONE OF THE TEST SIGNAL BOXES.....	24
FIGURE 10. INITIALIZING FANUC ROBOTICS ACTIVE X OBJECTS IN LABVIEW.....	26
FIGURE 11. REFERENCE NUMBER	27
FIGURE 12. TWO ADD OPERATIONS	28
FIGURE 13. DATA DRIVEN EXECUTION.....	29
FIGURE 14. REFERENCE AND ERROR WIRES.....	29
FIGURE 15. TWO SEQUENTIAL MOVE TO SUBVIS USED IN A ROBOT PROGRAM.....	31
FIGURE 16. BEGINNING PART OF MOVE TO SUBVI.....	32
FIGURE 17. SECOND PART OF MOVE TO SUBVI.....	34
FIGURE 18. LAST PART OF MOVE TO SUBVI.....	35
FIGURE 19. USAGE OF THE MOVE TO OFFSET SUBVI ON ROBOT PROGRAM.....	35
FIGURE 20. MATS CUSTOMISED LABVIEW ICONS.....	36
FIGURE 21. TEACH POSITION PROGRAMS LABVIEW FRONT PANEL VIEW.....	37
FIGURE 22. EXAMPLE OF OPENED POSITION FILE.....	39
FIGURE 23. A SCREENSHOT FROM MATS USER PANEL.....	40
FIGURE 24. EXAMPLE USAGE OF REG EVENT CALLBACK.....	42
FIGURE 25. USAGE OF THE TOOL IDENTIFICATION SUBVI.....	43
FIGURE 26. LABVIEW PROJECT EXPLORER WINDOW.....	44
FIGURE 27. NETWORK CONNECTION.....	45
FIGURE 28. COLLECTION OF CASSIDIA FINLAND'S TETRA PRODUCTS	47
FIGURE 29. POSITION SEQUENCE FLOW CHART OF EXAMPLE TEST CASE.....	48
FIGURE 30. FANUC ROBOTICS TEACHING PENDANT.....	51
FIGURE 31. FIRST PART OF THE EXAMPLE ROBOT PROGRAM.....	52
FIGURE 32. SECOND PART OF THE EXAMPLE ROBOT PROGRAM.....	53
FIGURE 33. LAST PART OF EXAMPLE ROBOT PROGRAM.....	54

TABLES

TABLE 1. Example test specification template.....49

APPREVIATIONS

Appreviation	Explanation
MATS	Mechanics Automated Test System. Automatic test development system that this thesis describes.
Host Computer	Computer connected to robot controller. Runs test programs and monitors the state of the robot.
PCDK	PC Developers Kit created by FANUC Robotics. Collection of ActiveX components that allow controlling of the robot with PC.
R-J3 robot controller	Robot controller developed by FANUC Robotics. Handles motion calculations, IO operations and communications.
LR Mate 200i robot	Industrial robot developed by FANUC Robotics.
SubVI	LabVIEW sub routine
DUT	Device Under Test

1 INTRODUCTION

1.1 General

Cassidian Finland has strict requirements for their products functionality, wear resistance and endurance. Various tests are performed during the development to verify the requirements. This way most of the product manufacturing and design faults can be detected before the products are released. When performed by employees these product tests would consume a large amount of work hours and they are difficult to repeat similarly. Results may vary depending on the test person, the test person's state of mind and test tools. For this reason Cassidian Finland started a project to create a test development environment for creating and executing mechanical endurance and wear resistance tests.

This thesis reports the software development and creation part of a mechanical test system called Mechanics Automated Test System or MATS. The main parts of MATS are FANUC Robotics LR Mate 200i industrial robot, FANUC Robotics R-J3 robot controller with IO modules, LabJack U6 data acquisition card and a host computer. MATS system block diagram can be seen in figure 1 and appendix 1.

*) User operable interface for additional I/O signals (FB2)
 **) User operable interface for analog measurement signals (FB3)

System block diagram

Monday, June 28, 2010

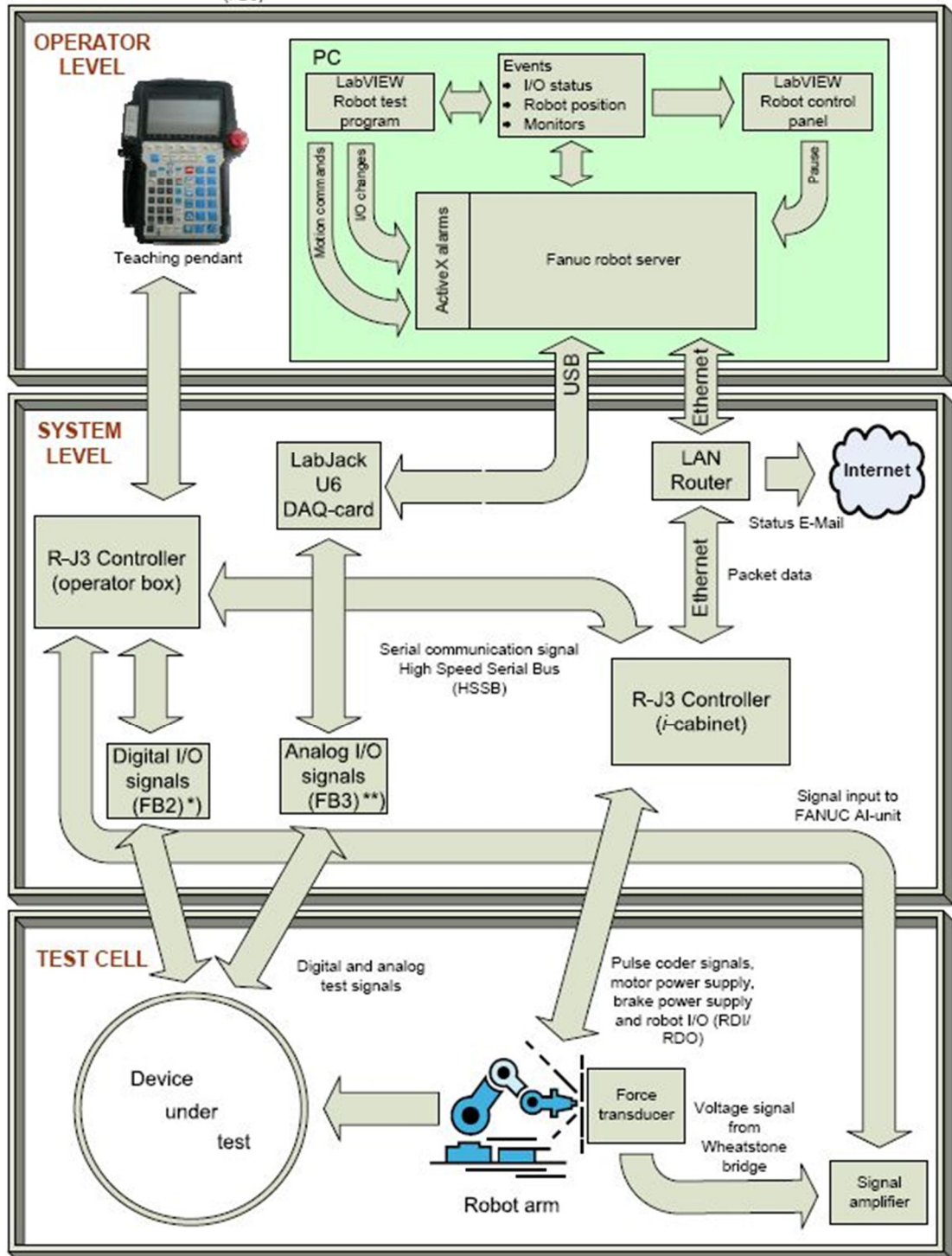


FIGURE 1. MATS block diagram

1.2 Goals

In the early requirement of the Cassidian Finland's Mechanics Automated Test System it is defined that the system must be capable of performing tests for Cassidian Finland's products for long periods of time and as autonomously as possible. This system could be used through the life cycle of the products from concept selection to confirming faults that arise from the field. The main products of Cassidian Finland in Jyväskylä are in the area of Professional Mobile Radios or PMRs. The company designs both hand held radio units and mobile vehicle radios.

To create a highly adaptable system without fully knowing what kind of tests it will perform, the best option is to use a programmable industrial robot. In general, robots have higher reach and are more versatile than pneumatic manipulators. By using an industrial robot as a device that manipulates the test objects, it is almost certain that most of the upcoming future products can be tested with the same test system, making only small modifications and additions.

For this purpose, Cassidian Finland had acquired two used FANUC Robotics LR Mate 200i robots and R-J3 controllers with various parts and pneumatic equipment.

The goal of the project was to create an adaptable and easy to use test creation and execution system utilising one of the robots. When building a system, available parts and programs should be used as much as possible. Using already purchased parts would keep the costs reasonable. When defining the goals of the project these three goals rose above the others.

- 1) Create automated test creation environment for mechanical reliability testing that utilizes an industrial robot.**
- 2) System must be as easy to learn and to use as possible.**

3) System must be highly adaptable for future EADS products.

The first goal means that test designers and operators should not need to have extensive programming or robotic experience to create or run tests. If the tests take too much time to create, the costs of test creation are too high for the system to be used as regularly as intended. Some level of technical or mechanical test background can be required from the persons who create the tests.

The first goal was the main concern in the software development part of MATS project. This required designing and creation of the system tools and process in a way that the test creation would be extremely easy and fast. Additionally, the test running environment and user interface for robot monitoring would have to be simple and user friendly.

The second goal was the adaptability of the system. This meant that the system needed to accept different kind of products, including future products that have not yet been designed. This was the major concern of the hardware design part of this project. This goal had to be taken into consideration when designing product attachments and robot grippers. In the software side there should be easy an access to any new devices that would be connected to the system.

All in all, this project was quite different from traditional robot applications. Usually industrial robots perform well defined pre-programmed task for long periods of time without any need of reprogramming. The objects that robots normally manipulate in production lines might stay the same for years. In this project the task of the robot was almost opposite. Every new fault or irregularity in the product would require a new robot program to be created. The robot also had to be able to manipulate objects that did not yet exist.

1.3 Topicality and importance

The current trend in the robot development is the replacement of separate

robot controllers with regular personal computers. In the near future robot controllers will most likely be replaced by normal office PCs that run real time and multitasking operation systems. One example of this kind of robot application is the border monitoring robot developed by VTT (Technical Research Centre of Finland). This robot has military grade PC running real time Linux operating system that handles all low- and high level controlling and communication tasks (Jalovaara 2010, 6). The reason for using PCs over robot controllers is obvious. PCs are cheaper, more adaptable and have more universal connection methods than traditional robot controllers (Robotiikka 1999, 34).

This thesis is a halfway experiment for a computer controlled robot. Even though the robot controller still handles all motion calculations and some of the IO operations, robot programs are created and executed with a regular office computer. A PC sends commands to the robot controller and then monitors their execution. An actual robot program is running in the computer's memory, not in the robot controller's memory.

Robot manufacturers search for new methods of creating robot programs. Robot programming tools are being developed to be more intuitive and easy to use. The results of this thesis show that graphical programming environment is highly effective in robot programming, saving time and money from program development.

1.4 Company details

Cassidian is a global provider of defense and information systems. Their repertoire includes military and paramilitary aerial, naval and land systems. Cassidian's products include for example cyber security, secure communication systems, weapon systems, services and support. In the year 2009 Cassidian had around 28.000 employees world wide and achieved revenues of 5.4 billion euros. Cassidian is a part of European Aeronautics Defense and Space Company also known as EADS (Cassidian website, 2011).

Cassidian is divided into integrated business units. Cassidian Systems con-

centrate to provide their military and paramilitary customers comprehensive and tailored security and communications systems. Cassidian Systems provides integrated solutions covering all domains (underwater, surface, shore, air, space) for naval warfare and maritime security. (Cassidian website, 2011).

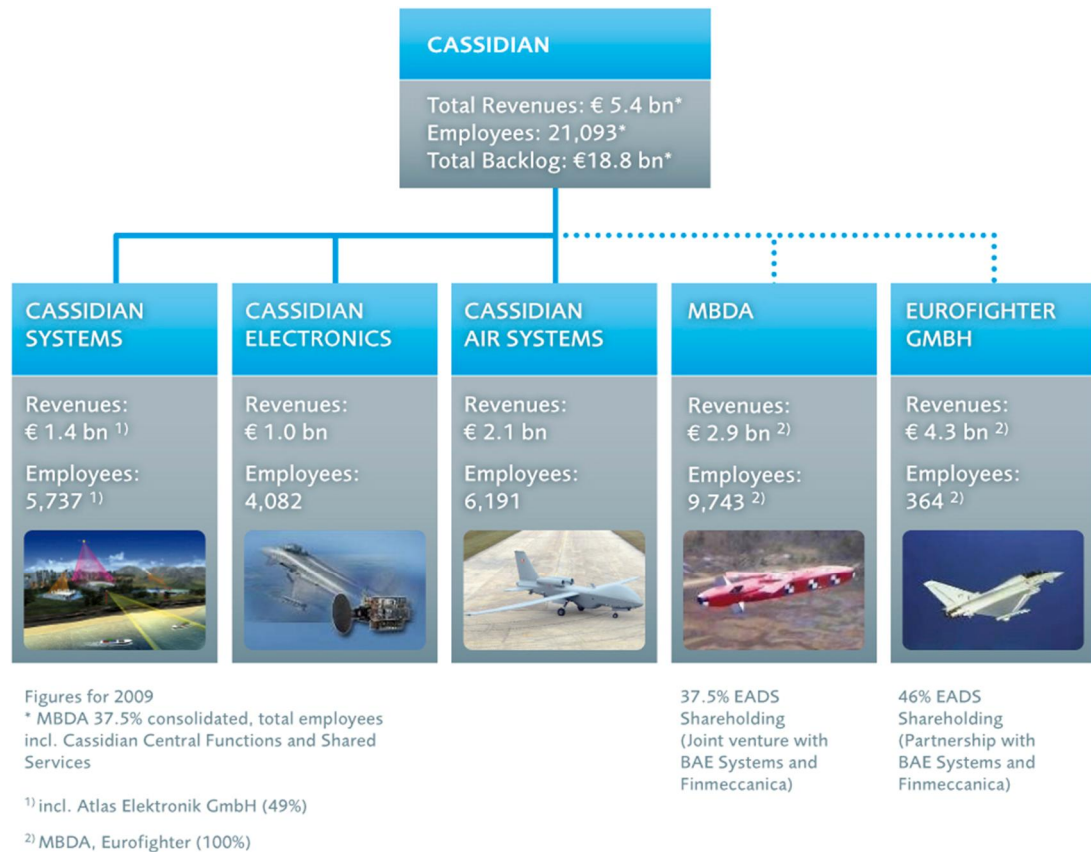


FIGURE 2. Cassidian organization chart.

In 2005 EADS Defense & Security bought Nokia Plc's professional mobile radio business that designed and manufactured professional radios and network devices. This new EADS unit was named EADS Secure Networks. The name was changed into Cassidian Finland in October of 2010. (EADS Defence & Security changes its name to Cassidian 2010).

Currently Cassidian Finland departments in provide professional mobile radios, network solutions and supporting applications and services. In year 2008 Cassidian Finland had 297 employees and a turnover of 156 million euros (Talouselämä 500 website). Cassidian Finland has two locations in Finland. These sites are located in Helsinki and Jyväskylä. This thesis was assigned by

Cassidian Finland in Jyväskylä.

2 THEORETICAL BACKGROUND

2.1 Robot programming

When industrial robots were first introduced, programming of the robots was handled with electromechanical switches. When a robot drove to a defined position, it activated the switch inside robot joints and the robot performed the next action. This was quite a primitive way to perform motion paths and changing the motion patterns required major changes to the robot's hardware configuration (Robotiikka 1999, 78.)

Another method to teach paths for robots was a leading method. The robot joints had pulse coders, and the robot arm was lead along the desired path by hand. This path was recorded to tape drive, and when that tape was played, the robot repeated the taught path. This was not very accurate or effective way to teach paths. Storing and changing of the tapes was difficult and time consuming (Robotiikka 1999, 78.)

Currently the most common method to create paths for robots is to save robot's positions inside virtual coordinates to controller's memory, and give motion commands from the robot's current location to one of the saved positions. Changing of the robot position is not done anymore leading by hand, but robots are driven to wanted position with remote control-like devices (Robotiikka 1999, 78.)

Now the revolution of the robotics is emerging. High speed of communication lines, accuracy of sensors, calculation power of the CPUs and large data storage capabilities are revolutionizing the minds and brains of the robots. They can react to things they "see" in real time and make decisions from the data stored in neural networks (Brooks 2002, 5).

2.2 LabVIEW programming

LabVIEW is a graphical programming environment developed by National Instruments. It was first introduced in 1986 for Apple Macintosh computers. In 1992 LabVIEW became available for other platforms than Macintosh. Now LabVIEW can be used with Windows, Unix, Linux and naturally Macintosh operating systems (Charterhouse Solutions website 2011).

LabVIEW is mainly used in engineering as an instrumentation, measurement and control tool but it can also be used as a higher level graphical programming environment. With LabVIEW, the programmer does not have to follow strict syntax of traditional programming languages. LabVIEW uses a programming language called G. G language is a data flow code and with it programmers can build applications by connecting program blocks with wires that represent different variables, values and constants (Travis & Kring 2006, xxxi).

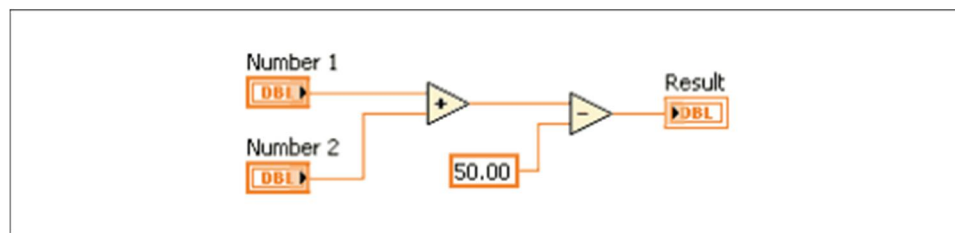


FIGURE 3. Example of data flow programming.

LabVIEW is not just instrumentation and programming environment but it also assists test designers by visualising data types and errors with colours and symbols. This is why learning curve in LabVIEW is not as steep as in traditional programming languages. An inexperienced program developer can have a functional and executable application with few minutes of learning. The down side of LabVIEW is its price. Other programming languages have lots of free development tools, but LabVIEW is only usable if purchased.

As with all higher level programming languages, program execution performance is somewhat lower with LabVIEW than with more basic programming languages like C++ or Visual Basic. (Travis & Kring 2006, xxxi).

LabVIEW programs consist of two different layers, block diagram and front panel. The front panel is the window that opens when a new LabVIEW program is created. It contains the controls that the user interacts with and indicators that give output from process or program.

Block diagram is the window where LabVIEW test designers build their applications. It holds the graphical representation of the data flow code. In block diagram the application is made by creating and connecting terminals, nodes, and wires.

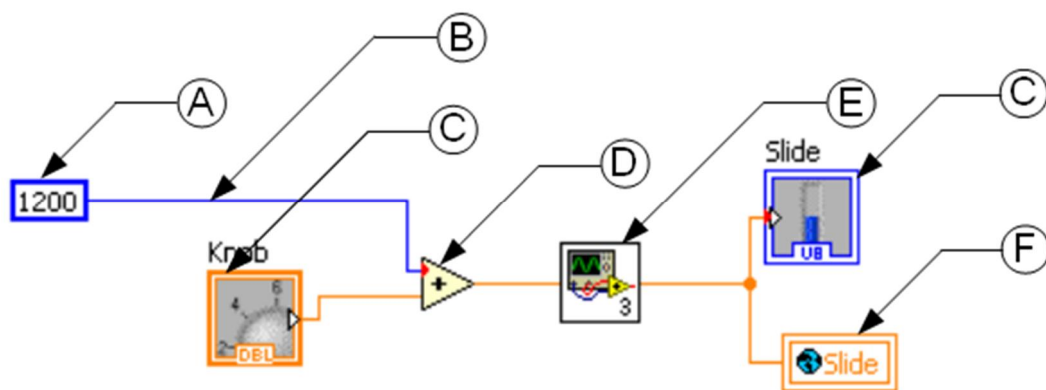


FIGURE 4. LabVIEW objects

The terminals (C) can be used to transfer data between block diagram and front panel objects. Node (D) is a block diagram object that performs actions with variables and constants (A). Nodes represent the operators, functions and subroutines in line based programming environments. All the LabVIEW block diagram objects are connected with wires (B). The colours of the wires represent the data it relays. This makes LabVIEW code easy to read and interpret.

When a program is built with different graphical elements and wires, a complex program might start to look like a bowl of spaghetti carbonara more than a functional computer program. This is a big down side of LabVIEW environment. LabVIEW subVIs (E) and global variables (F) can help LabVIEW program to look clearer.

subVIs or LabVIEW sub-routines are large program segments that appear in block diagram to be a small nodes. SubVI is a call method of LabVIEW environment. SubVIs can have input and output terminals, and also front panel objects. Global variables are data objects, that can be used to pass data between several simultaneously running LabVIEW programs (NI LabVIEW Environment Basics).

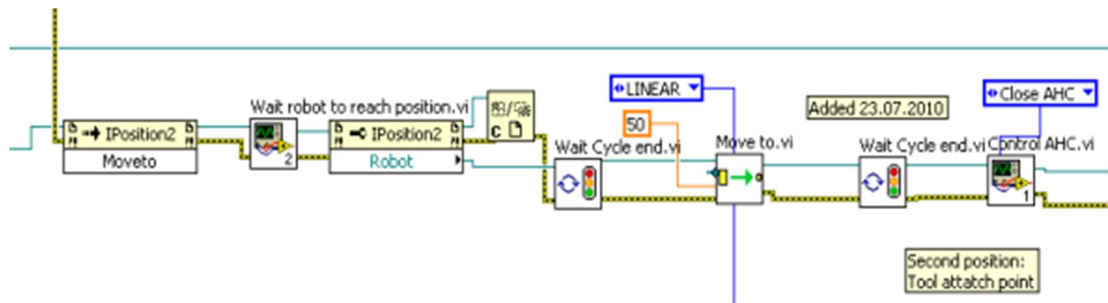


FIGURE 5. Example from MATS block diagram.

2.3 ActiveX

ActiveX is a universal object interface introduced by Microsoft in year 1996. ActiveX is a common name for OLE controls and Component object model (COM). In MATS ActiveX objects are used to access the robot's functions and methods. FANUC Robotics has created an ActiveX library that utilises the robot's functions via a network interface.

2.4 MATS Software components

2.4.1 General

One main goal of the application design of MATS was that the system must be easy to use and the robot programs easy to create. Often robot programming tools and languages can be quite complicated. Usually robot programming is done either locally with teaching pendant or with an external offline programming tool that runs on a computer. Offline programming of the LR Mate 200i robots and R-J3 controllers are usually done with FANUC Robotics offline programming software called WinTPE.

These programming methods are highly effective in most cases where a robot program's stays unchanged for long periods of time. For MATS, Cassidian Finland wanted an even more straightforward solution. The test programs must be created and managed with an easy to use computer based application. Additionally robot monitoring must be handled with a computer. Following sections explain about selection and interaction of software components that were used with Mechanics Automated Test System.

The system where MATS robot programs are run is a normal office desktop computer running Microsoft Windows XP professional.

2.4.2 FANUC Robotics PC Developers kit

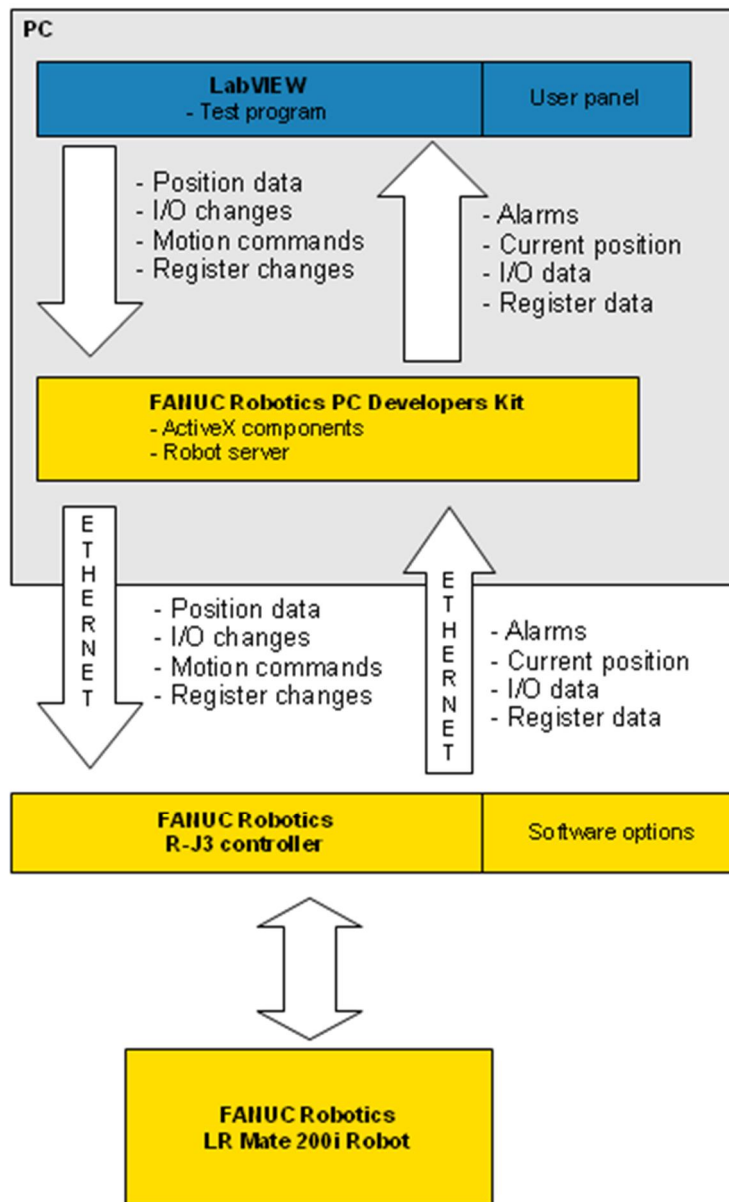


FIGURE 6. Block diagram of MATS software components

FANUC Robotics PC developer's kit (PCDK) is used as communications software between the robot and a computer. PCDK is actually a collection of ActiveX components, their documentation and supporting computer applications. PCDKs ActiveX objects allows the host computer to send commands through the robot server. PCDK package also has example programs written in Visual Basic programming language.

PCDK allows the computer to read the desired register and digital signal values from the robot controller's memory in to the host computer. These values can be used in any program that requested the information. It can also change values and issue commands through ActiveX controls. PCDK is currently somewhat old-dated program for these kinds of tasks, but R-J3 robot controller does not support any newer communication applications. Newer FANUC Robotics robot controllers have better support for computer integration than what R-J3 and PCDK offer.

PCDK was the only reasonable software choice to be used for robot/PC communications in our projects time frame. Another option would have been to create this kind of ActiveX library from scratch, but it would have taken years of reverse engineering and software developing. The datasheet of the PCDK can be seen in appendix 4.

2.5 Robot server

Robot server is a computer program developed by FANUC Robotics and it is a part of the PCDK software package. Robot server is an ActiveX executable program and enables the robot connection and relays commands from ActiveX components to the robot controller. It initiates the connection with TCP/IP protocol and usually operates through Ethernet connection. Computer side communications are handled through ActiveX object interface. ActiveX object interface allows robot test designers to use any programming tool that suits best for their purpose (PC Developers Kit 2011).

2.5.1 LabVIEW in Mechanics Automated Test System

National Instruments LabVIEW was selected as the main test creation software for Mechanics Automated Test System (MATS). Cassidian Finland already had a program license for LabVIEW and employees who had LabVIEW programming experience. The idea of building robot programs graphically was also alluring. The LabVIEW version used for MATS was LabVIEW 8.5.

FASTEMS Plc the Finnish importer of FANUC Robotics did not have any experience or knowledge of any applications where robot programs were created with LabVIEW. They said that it should be possible if LabVIEW supported ActiveX components. FASTEMS provided computer side software PC Developer's Kit and controller side software called Data communications option. Together these allowed robot programs to be created with computer and with any programming tool that supported ActiveX components.

LabVIEW has had a full support for ActiveX components since version 8.2. LabVIEW can act as an ActiveX client to access ActiveX objects, properties, methods and events (Using ActiveX with LabVIEW 2006).

With the combined features of LabVIEW and PCDK it was possible to create all needed tools under LabVIEW programming environment. FANUC Robotics has a ready made computer programs that can act as a user panel or programming environment; however, these programs cost money and test creation would not be as easy as it is with LabVIEW. Using one platform for robot programming, monitoring and project management would make the usage of the system much easier than using large collection of pre-made applications.

2.5.2 Data communications option

Robot controller software that was needed to enable robot/PC communication is called data communications option. Data communications option is installed on robot controller with PCMCIA card. FANUC Robotics refers robot side software packages as options, and they must be purchased separately. Data communications option enables the controller to send and receive packages between computer and robot controller. The contents and consistency of these packages remain confidential information that FANUC robotics is not revealing in any of the PCDK related documents.

It is clear that this data communications option installs at least two programs to robot controller. One sends data to PC and the other one receives data from PC. The receiving program has some control over robot, because it can

issue motion commands, change signal values and change robot configuration.

2.5.3 Robot controller alarms

When the robot controller detects an unwanted state it raises an alarm. Severe alarms abort current program cycle and notify user with error code and message. Alarms are displayed in the Teaching Pendant's alarm menu, and they can also be sent to host computer via Ethernet connection.

In MATS ten most recent alarms are displayed in user panel and new alarms are sent to operator via email. The operator is notified with alarm time, error code, message and test programs current status. This way operator can immediately react to error situations.

2.6 MATS Hardware components

2.6.1 General

This section introduces the hardware components of MATS so that the readers of this thesis have a better understanding of the test environment and its software components. All hardware in the system is controlled or monitored with customized computer applications. Therefore it is important to know the basics of the devices that are controlled and monitored.

2.6.2 FANUC Robotics LR Mate 200i industrial robot



FIGURE 7. FANUC LR Mate 200i industrial robot.

The robot FANUC LR-Mate 200i that Cassidian Finland acquired for test system suits this kind of work well. It is an articulated robot with six degrees of freedom. It has a modular construction and an electric-servo driven table top robot with a small footprint. The six degrees of freedom ensure that the robot is versatile enough to manipulate small buttons and parts that are in different angle compared to the main body of the product being tested.

The robot's maximum speed for the sixth joint is $480^{\circ}/\text{sec}$. The speed of the robot is limited to $250\text{mm}/\text{sec}$ which is the safety limit defined by an international standard. The robot's reach is 700mm , and this is enough for about ten or more products to be tested in one session, depending on the product's size. Usually the products being tested are a bit larger than a modern mobile

phone.

The maximum joint payload is 3kg. The robot is not powerful enough to perform tests that need high forces, like fall tests or professional mobile radio's hull endurance tests, but is enough for testing smaller parts, like keys, keyboards and battery releasing.

The robots repeatability is up to 0.04mm. The robot is accurate enough to locate buttons and other small mechanical parts. The datasheet of the LR Mate 200i robot is in appendix 2 (LR Mate 200i datasheet, 1999).

2.6.3 FANUC Robotics R-J3 Robot controller



FIGURE 8. FANUC Robotics R-J3 robot controller.

Robot controller is the brain that moves the hand that is the actual robot. For MATS FANUC Robotics R-J3 controller has all required features. It has 32-bit

dual process architecture. The processors handle motion calculations and communication processes independently. It can be modified and expanded by buying software options and I/O modules. The controller also has an in-built Ethernet card that enables easy host computer access and FTP server for backup operations.

Positions of FANUC Robotics LR mate 200i robot must be taught with teaching pendant. FANUC Robotics R-J3 controller can store only a limited amount of positions so it was anticipated that positions had to be uploaded to LR Mate 200i robot controller from external source, most likely from host computer. The R-J3 controller has an inbuilt Ethernet connection and FTP server. These were earlier used to make system backups and for saving robot programs to R-J3 controller's memory. The datasheet of the R-J3 controller is in appendix 3.

2.6.4 FANUC input and output modules

FANUC R-J3 controller has a modular input/output interface module model number BIF04A1. This unit can hold up to five input and output modules that can be both digital and analogue. These modules can be changed according to the requirements of the system. When Cassidian Finland purchased the robot and the controller there was two digital input modules (AID16D) and two digital output modules (AOD16D). One analogue input module (AAD04A) was purchased to read the signal from HBM force transducer that is installed on the head of the robot hand.

2.6.5 SMC Auto Hand Change System

MATS is equipped with SMC MA310 Auto Hand Change (AHC) system. This AHC system can change robot tools to accommodate to different tasks. With AHC system the robot can change tools autonomously whenever the robot program requests that.

There are five tool stands inside the robot work area that can be used to hold tools that the robot uses during a test session. SubVIs for picking up a tool

from tool stand, and leaving tool to empty tool stand had to be created. AHC system has six pneumatic ways and twelve signal ways that initiate connection to every tool hand that is picked up.

2.6.6 LabJack U6 data acquisition card



FIGURE 9. LabJackU6 and one of the test signal boxes

LabJack U6 is a data acquisition card with digital and analogue signal ways. LabJack U6 data acquisition card was acquired as a modifiable signal reader to read digital and analogue signals from devices under test. These signals can include for example connection signal of a key, or the resistance of a battery. Some of the LabJack signals are connected to the test signal boxes which allow a tester to easily connect needed test signal lines to the device

under test.

LabJack U6 was shipped with Windows driver and LabVIEW example programs. This enabled easy computer access via USB cable and seamless LabVIEW integration. This data acquisition card has good connectivity and measurement options. (LabJack U6 (-Pro) User's Guide, 2009).

2.6.7 HBM AE101 50N force transducer

HBM S2/50N Force transducer is mounted directly under the robot arm. The force transducer can be used to measure pushing and pulling forces up to 5kg (HBM S2 Force Transducers data sheet). LabVIEW applications could also use force information to identify objects by picking up an object and measuring tensile force. The signal from force transducer is amplified by HBM AE101 industrial DC amplifier. The amplified signal is then sent to FANUC analogue input unit. This analogue signal information is transformed to integer number in range from -2047 to + 2048. This value can be read to LabVIEW through custom made subVI. This subVI also converts integer pulse value to grams.

2.6.8 SMC ZSE40 pressure switch

SMC pressure switch monitors the pressure of two pneumatic suction lines. When using suction caps to pick up objects the pressure switch can be used to confirm successful pick up operation. SMC ZSE40 pressure switch monitors two pre set pressure limits and turns two digital outputs on or off depending on the device settings. Test developers can react to these value changes and program the robot by trying unsuccessful pick up operation again or by notifying operator with email about the failed pick up operation.

3 MECHANICS AUTOMATED TEST SYSTEM

3.1 Program architecture

3.1.1 Initializing and connecting the robot object

LabVIEW accesses ActiveX objects of FANUC Robotics PCDK through automation reference number control or the ActiveX container. Both of them are also LabVIEW front panel objects (Using ActiveX with LabVIEW, 2006).

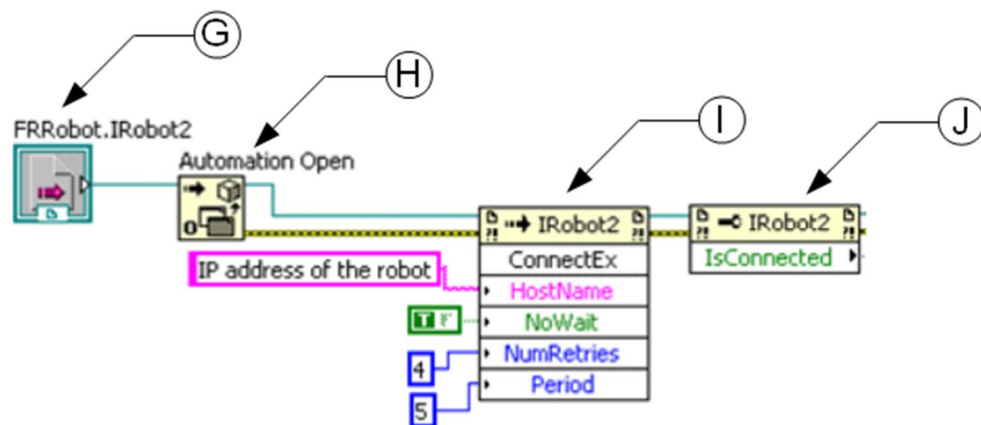


FIGURE 10. Initializing FANUC Robotics ActiveX objects in LabVIEW.

In MATS, the automation reference number is used to access ActiveX objects. When LabVIEW function *Automation Open* (H) is called for *FRRobot.IRobot* object (G), it initializes all ActiveX methods (I) and properties (J). Opening *FRRobot.IRobot* ActiveX object causes the robot server to start automatically. The robot server handles communications between computer and robot. The robot server runs silently in the background while robot programs are being executed, and then shuts down when *FRRobot.IRobot* object is closed.

The first action after opening an ActiveX object is connecting the robot server to the robot. This can be completed easily with ActiveX methods *Connect* or *ConnectEx* (I). *ConnectEx* method gives more options about connection times and number of retries. After the robot connection is initiated all robot controller

data can be accessed with same kind of line notation that is shown in figure 10. LabVIEW offers an easy way to explore ActiveX object's properties and methods. When the robot reference number is clicked with the right mouse button, a menu opens where there is option called create. Here the test designer can examine all properties and methods that exist in the selected object. Despite easy exploring of methods and properties, it still is a huge task to build all of these as LabVIEW sub-programs. PCDK has dozens of properties that all can contain hundreds of methods and properties.

3.1.2 Reference number

In LabVIEW a reference number is used to identify objects from each other. Reference is a random number that LabVIEW creates each time the object is called. LabVIEW uses reference number to identify which instance of the object is being used. With PCDK reference number represents the robot object that is opened when *FRRobot.IRobot* is initialized. Through the robot object test creator can access PCDK's properties and methods. The reference number is passed between LabVIEW terminals and nodes with a wire, like any variable.

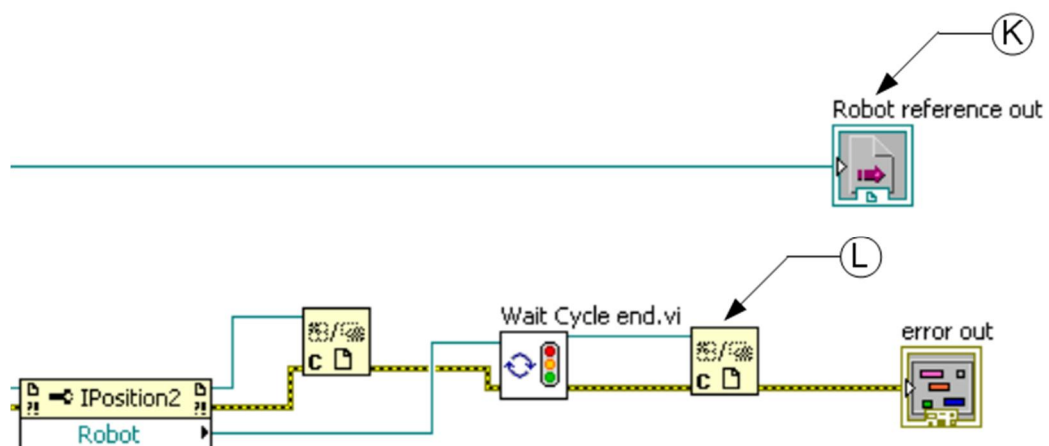


FIGURE 11. Reference number

The same robot reference number runs through the whole LabVIEW main program. If *FRRobot.Irobot* instance is cloned in sub-programs, this cloned

reference number is closed inside a sub-program (L), and the original reference number is passed to the next sub-program (K). This keeps the LabVIEW loop conditions functional, and the program development clearer. If the original robot reference number were closed inside any loop, the whole LabVIEW program would crash when the loop would be to run another cycle.

3.1.3 Program execution order

LabVIEW is a graphical programming environment and the program execution order is not always as clear as in traditional line based programming languages. LabVIEW terminals and subVIs are executed from left to right. If some functions are at the same horizontal level but one is on top of the other, these functions are tried to be run simultaneously. (Block Diagram Data Flow 2006).

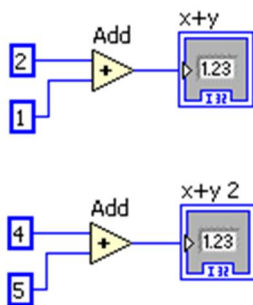


FIGURE 12. Two add operations

In figure 12 two add operations are executed simultaneously. This is useful in some applications where simultaneously performed functions will not affect physical world. In the case of robot software development, the correct execution order is very important, for example when a robot is commanded to move to a certain position and perform an action when reached there. If these two imaginative sub-programs are located in same horizontal level, both movement and control commands are executed simultaneously. This causes the robot to fail in completing its task.

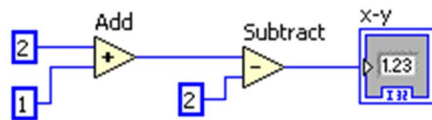


FIGURE 13. Data driven execution

LabVIEW execution order can also be called data driven. Program blocks are run when they have all required data. In figure 13 subtract action is not performed before add operation gives data to it.

With robots, it is important that the first move is completed before trying to execute the next one. With PCDK sending a new command to the robot controller that is still executing a previous task causes a system crash. PCDK has good functions to ensure that the robot has reached its destination before continuing the program cycle. This situation is achieved by an *IsAtCurPos* method placed inside a loop, where methods output Boolean is connected as a loops stop condition.

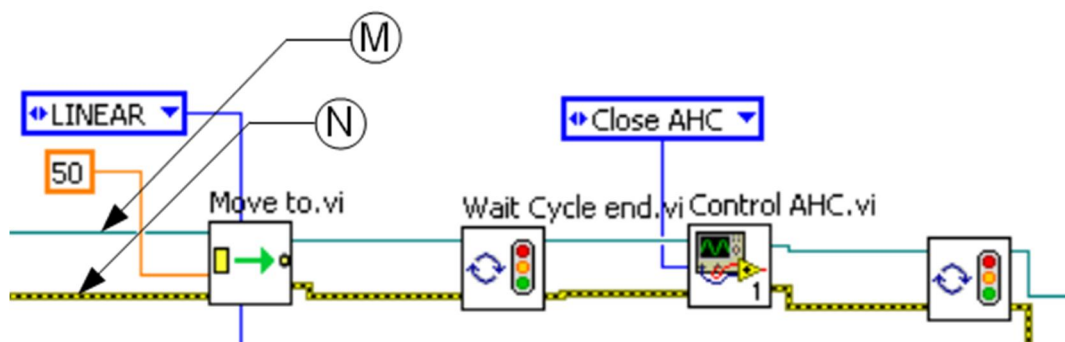


FIGURE 14. Reference and error wires

There is also a way to ensure that any method or property is not executed when the system is in error state. This is done by connecting the same error wire (N) to every block and sub-program in LabVIEW program. This single error wire also makes the program execution linear, since all LabVIEW functions wait for the error data before execution if the wire is connected. In MATS the program execution order is managed with error wire (N) and the robot ref-

erence number (M).

3.2 Sub programs

3.2.1 General

MATS includes 45 subVIs that can be used for the program creation. These subVIs include motion commands, position reading, tool changing, email sending etc. Overall amount of the used LabVIEW functions is hard to determine. The estimated amount of the used functions is in the range of 1000 - 1500.

Using and initializing all needed PCDK functions with ActiveX interface in LabVIEW is somewhat time consuming. If robot programs would have to be created every time from scratch, it would not be efficient or cost-effective. To achieve the main goals of easy usability of the test creation system, a collection of easy to use subroutines had to be created. These subroutines could be used during the creation of the test programs.

Subroutines in LabVIEW are called subVIs. SubVIs can be called from any block diagram. When subVI is called it runs the LabVIEW program it refers to. (Creating subVIs 2008).

With MATS, almost every subVI has at least two input and output wires. These two wires are the Robot reference number and error wire. Both of them are run uninterrupted through the whole robot program. Reference and error path determine the program execution order and subVIs reserve these two values until they have performed their task. This prevents running two programs simultaneously.

3.2.2 Move to example subVI

Move to subVI is one of many subroutines created for MATS. This subroutine reads position data from a position file and moves the robot in a defined way

to these coordinates.

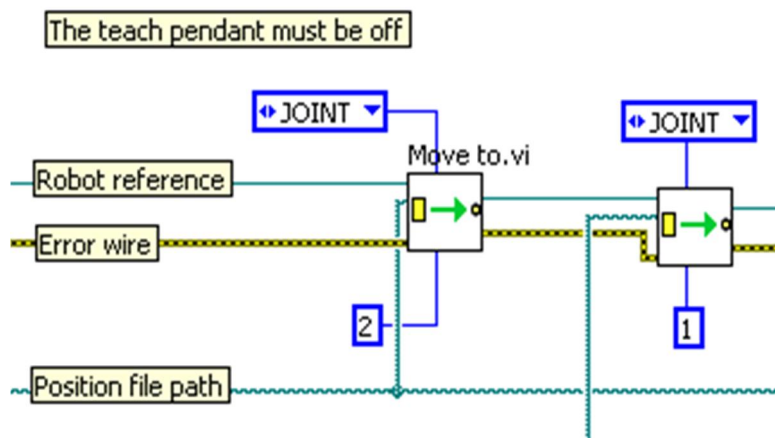


FIGURE 15. Two sequential "Move to" subVIs used in a robot program.

The function of this subroutine may sound extremely simple, but constructing this subroutine from PCDK ActiveX components is quite complicated. This LabVIEW program's block diagram is so large that it is represented in several parts.

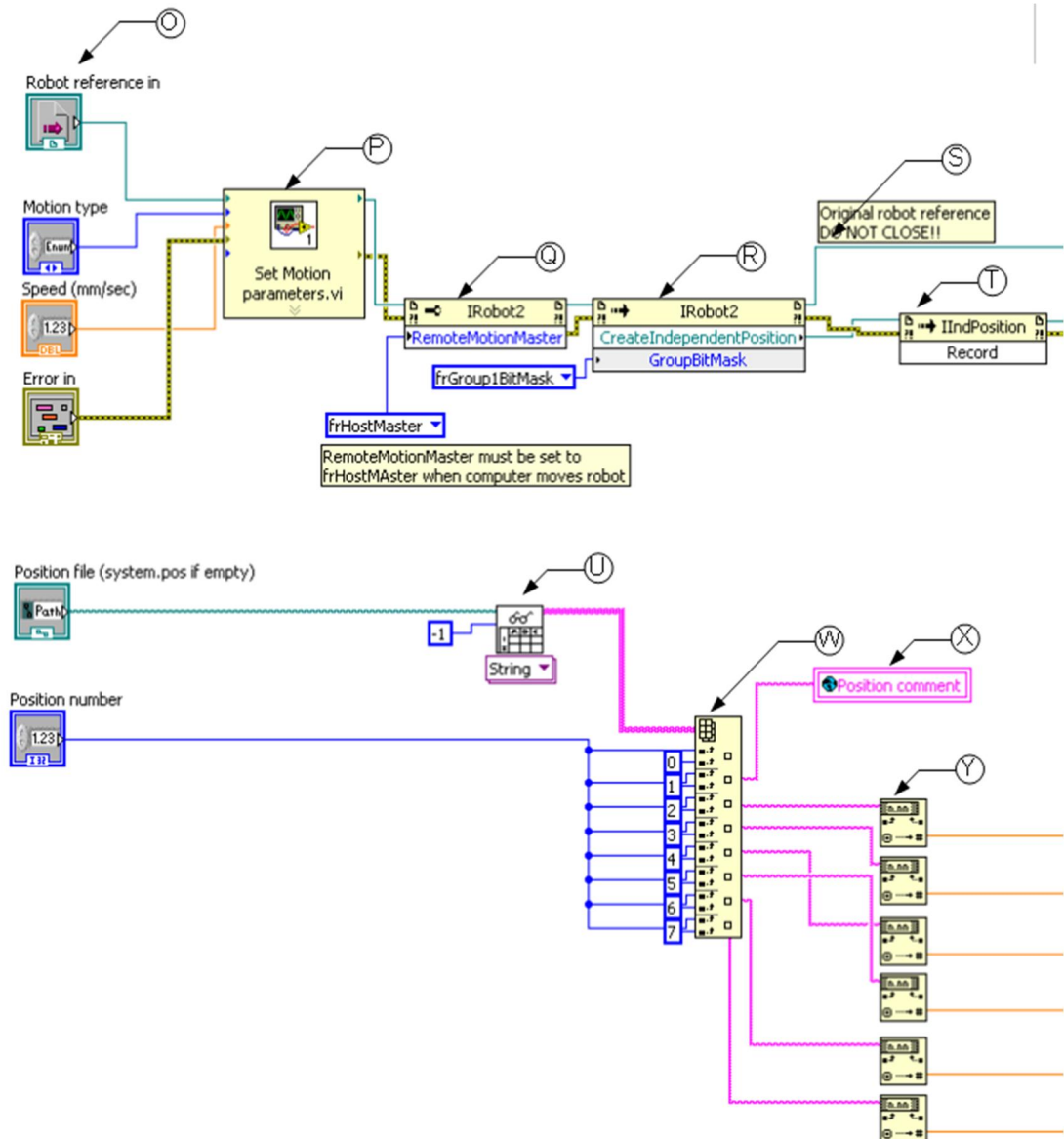


FIGURE 16. Beginning part of Move to subVI.

As seen in figure 15 in the the left side there are terminals that relay values from the main robot program to subVI (O). These values are the robot reference, motion type, speed and error wire. Position file path and position index number are also brought from main program. All these values are needed in subVI.

Inside *Move to* subVI is another subVI called *Set Motion parameters* (P). *Set Motion parameters* subVI change the robot controller's register values to set the next movement's type and speed. The values are reset automatically by the robot controller when the robot has reached the desired position.

RemoteMotionMaster (Q) property must be set to frHostMaster. This way the robot controller knows that the host computer can issue motion commands to robot controller.

Because the controller normally has all positions saved to its own memory, an independent position must be created inside the LabVIEW program. This is done by calling *CreateIndependentPosition* (R) method. This method creates a new empty independent position that can be used outside of the robot controller. For *CreateIndependentPosition* method a motion group must be defined. The robot controller can have many motion groups. These motion groups represent the robot's degrees of freedom. In MATS there is only group 1 in use.

Record (T) method is called for independent position. This ensures that the independent position has all the needed information that move to command can be issued to it. The record method also saves the current position of the robot, but this position data can be changed. Note also how original robot reference number (S) is transferred to reference output port.

Opening the position file is performed simultaneously with first part of Move to subVI. This does not cause error because reading from the file does not use robot server. First all positions from position file is opened as an array (U). Desired position from array is broken down to series of strings (W). Strings represents one of the elements in X,Y,Z,W,P,R coordinate system. Coordinate strings are then converted to numbers (Y) and transferred to *IXyzWpr* property. Position comment is transferred to user panel via global variable (X).

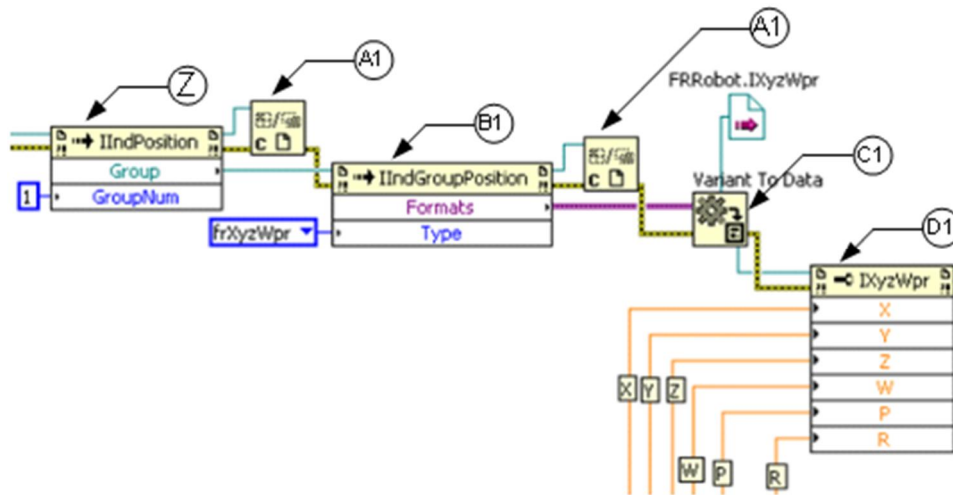


FIGURE 17. Second part of Move to subVI.

In second part of Move to SubVI the first motion group is selected for Independent position with *IIndPosition* method (Z). Then position data of the recorded independent position is formatted to X,Y,Z,W,P,R format with *IIndGroupPosition* method (B1). After this the independent position is type cast to the same format with *variant to data* node (C1). Then the position data of the independent position can be read and written. In this example the position data is written to the independent position with *IXyzWpr* property (D1).

The *Close reference* (A1) nodes that are placed after some of the methods and properties ensure that the opened instances of robot objects are closed when they are not needed any more. This saves computer resources and prevents problems that occur when multiple robot objects are alive in computer memory. Sometimes in these kinds of situations the robot server could not handle the commands sent to multiple robot objects, and caused an error message and program crash.

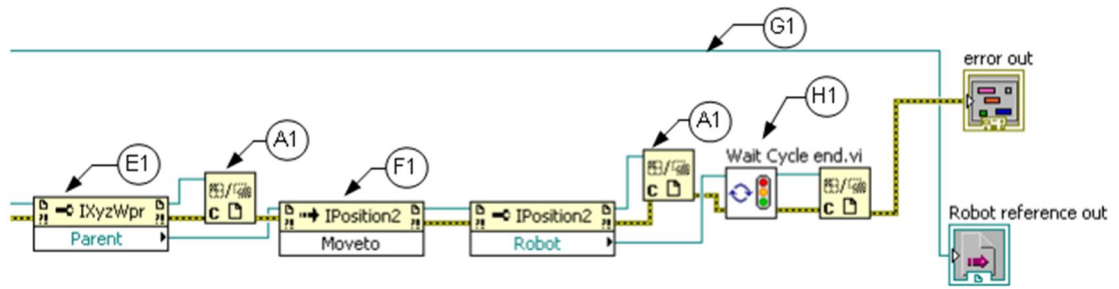


FIGURE 18. Last part of Move to subVI.

In last part of Move to subVI *Parent* property (E1) is called to gain access to the properties of the independent position. *Moveto* (F1) method is the command that sends the independent position to the robot controller and moves the robot. *Wait Cycle end* (H1) is another subVI that reserves the robot reference number and error line until the robot has reached the position. Note how original robot reference (G1) is passed to output terminal.

There is also a Move to offset subVI that changes positions according to offset information. Workbenches where the test objects are placed are crafted accurately, so with offset motion command the test designer can use the same program and positions in multiple places in the work area.

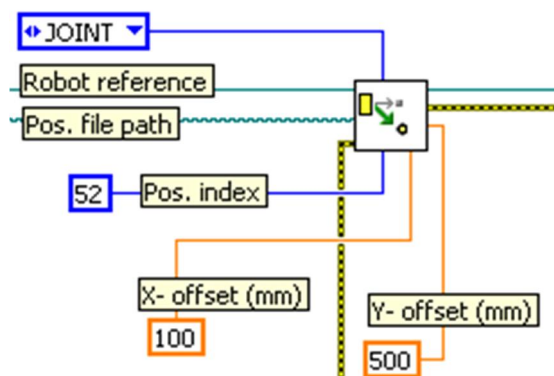


FIGURE 19. Usage of the Move to offset subVI on robot program.

3.3 LabVIEW subVI icons

To make program creation and execution easier, three hours were spent to create customized icons for most commonly used MATS subVIs. This made

LabVIEW robot programs much easier to read. National Instruments does not yet have a large collection of robot themed program icons. There were maybe two or three robot related icons in National Instruments website, but they really were not descriptive enough for this purpose. Instead there were many icons in other sections that could be used as a template for custom made icons.

Because LR Mate 200i is coloured bright yellow, this yellow colour was used in the icons to represent the robot. This clarified icons more and now test designer or any other person who has never worked with MATS can immediately gain some idea of functionality of a sub-program based on its name and icon.

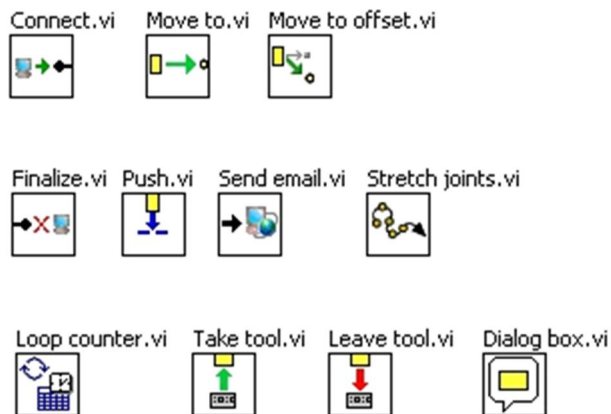


FIGURE 20. MATS customised LabVIEW icons

3.4 Position teaching

3.4.1 General

Teaching positions to a robot is something that cannot be avoided in current industrial robot systems. Sometimes position teaching can be somewhat cumbersome and time consuming. When LabVIEW was selected as a main program creation tool, it was clear that some kind of position teaching program had to be created. There was no ready implementation or template for robot position teaching program available for LabVIEW environment, from FANUC Robotics or National Instruments.

3.4.2 Teach positions LabVIEW program

Position teaching of MATS is handled with host computer and LabVIEW program named Teach positions. PCDK ActiveX components allow near real time position data to be read from R-J3 controller. LabVIEW can read this data and save it to file. With Teach positions LabVIEW program test designer can easily create new or open existing position files. There is no limit how many position files can be created, but one position file can contain -100 positions. This limitation is set only to keep position files manageably sized. One robot program can use data from any created position file.

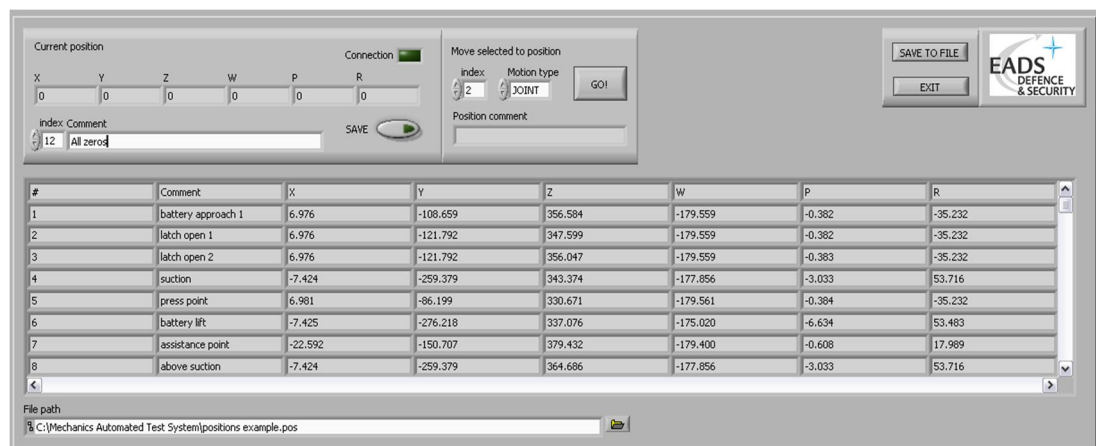


FIGURE 21. LabVIEW front panel view of the Teach positions program.

“Teach positions” LabVIEW program reads the current position from robot controller and saves it to file, when the operator so chooses to. All save, retouching and motion actions are controlled by the user with LabVIEW front panel objects. Positions are identified in the program code with path the of the position file, and index of the position in that file. The position comment is added to remind users about the function of a position.

Based on the results of early trial runs of position teaching it was decided to add the possibility to give motion commands to the robot from “Teach positions” program. This way the test designer can test the functionality of position combinations immediately after the positions have been taught. The test designer can also select the motion type of the motion to be tested. Robot has

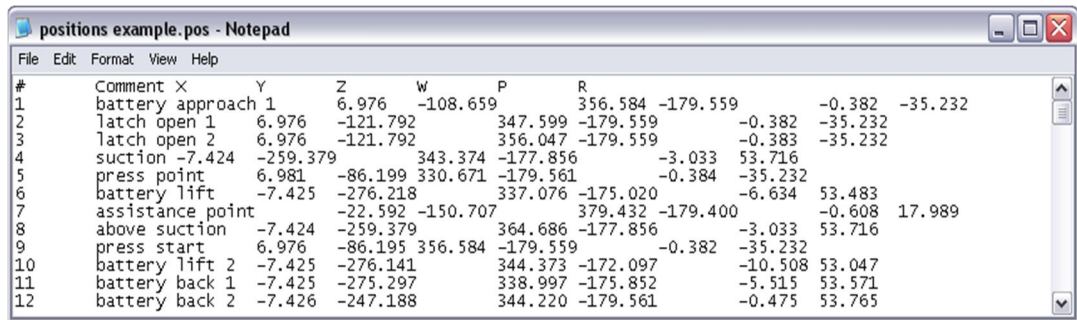
joint and linear motion type. Linear motion type travels path between two positions following a strict straight line. Joint motion type exchanges precision for speed. Joint motion makes transfer between two positions faster, but does not necessarily follow straight path.

The possibility of jogging the robot with LabVIEW program was also studied. Jogging LabVIEW program initiated connection and retrieved current position data from the robot controller. When this position data was changed with LabVIEW front panel object or computers keyboard, LabVIEW sent “move to” command to the robot. This experiment worked, but the delay caused by the Ethernet connection was too large for jogging with LabVIEW or keyboard to be allowed. Robot jogging has to be performed with the robot controller's remote control like device called the Teaching Pendant. The teaching pendant allows delay free and accurate control over robot.

3.4.3 Position format

Positions are saved to file as a plain numbers. It was decided to keep position data accessible, so an experienced test designer could change positions from the position file without full retouch procedure. Files that store positions have a postfix “pos”. The position file is actually a table where “tabulator” is used as a separator for columns and “enter” as a separator for rows. This way the position files can be opened with a compatible spread sheet program. Every position has X,Y,Z,W,P and R coordinate position information, index number from 1 to 100 and a comment. Values X,Y and Z express a 3D location of the robot tool inside workspace. Values W,P and R represent the tilt of the robot tool in that position.

The position data is saved as real numbers with three decimals. The index number of the position is used when creating robot programs, to identify different positions in the position file. The position comment helps the test designer to remember what the position actually is. The comment is also displayed in MATS user panel when a motion command to that particular position is issued.



#	Comment	X	Y	Z	W	P	R			
1	battery approach	1		6.976	-108.659		356.584	-179.559		-0.382 -35.232
2	latch open 1		6.976	-121.792		347.599	-179.559		-0.382	-35.232
3	latch open 2		6.976	-121.792		356.047	-179.559		-0.383	-35.232
4	suction	-7.424	-259.379		343.374	-177.856		-3.033	53.716	
5	press point		6.981	-86.199	330.671	-179.561		-0.384	-35.232	
6	battery lift		-7.425	-276.218		337.076	-175.020		-6.634	53.483
7	assistance point			-22.592	-150.707		379.432	-179.400		-0.608 17.989
8	above suction		-7.424	-259.379		364.686	-177.856		-3.033	53.716
9	press start		6.976	-86.195	356.584	-179.559		-0.382	-35.232	
10	battery lift 2		-7.425	-276.141		344.373	-172.097		-10.508	53.047
11	battery back 1		-7.425	-275.297		338.997	-175.852		-5.515	53.571
12	battery back 2		-7.426	-247.188		344.220	-179.561		-0.475	53.765

FIGURE 22. Example of opened position file.

3.5 Data collection

The tests that are performed with the system can collect a wide variety of different measurement data. The only limitation to the output data is the sensory equipment. Currently test developers can measure pressure and tensile forces with force transducer, digital and analogue output data with LabJack U6 data acquisition card and Fanuc IO modules. Test developers can also program tests in a way that the test is paused when a certain part of the test is completed. This way the operator can perform complex measurement actions manually if needed.

Data that is collected by LabVIEW test program can be processed with LabVIEW functions. Data can be saved to a file inside the host computer's memory or sent to an operator by email. LabVIEW has many inbuilt functions that can perform all file handling operations. Send Email subVI has an input port that can be used to send data to the operator. Email notifications can be turned off during the program testing. This way MATS does not spam the test creator's mail box if there are errors, or a test cycle is done multiple times.

3.6 User panel

Mats user panel provides information for a tester during the test program execution. MATS user panel also has a few buttons for the user to have some control over the robot. These buttons execute LabVIEW sub program for various purposes. Pause button interrupts the robot motion. Operators can use

this button to perform calibration actions inside robot work area safely. Continue button continues the program cycle after it is paused. Stop button aborts the program cycle and closes the opened robot connection.

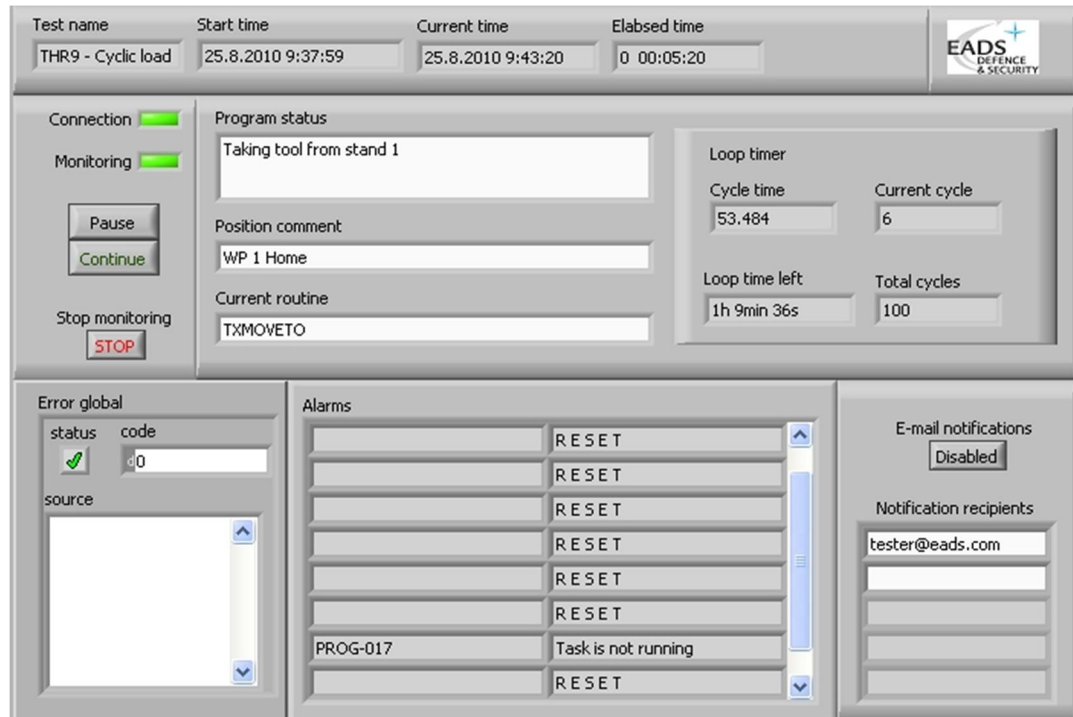


FIGURE 23. A Screenshot from MATS User panel.

Values of MATS User panel and running robot program are transferred through global variables. This way the test designer will not have to draw wires from Start mats user panel subVI block to every value instance that is being monitored or changed. Changing of the values is handled inside subVIs and changed values are updated to the user panel.

Great numbers of the test cases that are run with the system repeat the same cycle for tens of thousands of times. The data in loop timer boxes is linked to the subVI that has the same name. The loop timer box shows information about the duration and count of the loop, that test program is currently running. The loop time left box shows estimation how long the loops are going to run. The current cycle box shows how many cycles are already performed.

The topmost information box shows general information of the test program. It

shows the test program's name, test start time, system's time, and elapsed time of the test.

Program status data field shows the information about the current state of the test program. Position comment field shows the comment of the position, where the robot is moving if this data has been saved when the position has been taught. Current routine shows the name of the routine that the robot is performing. Alarms table shows the ten most recent alarms from the robot controller. New alarms are also always sent to the operator via email.

The robot's status changes are being handled by activating PCDK ActiveX event monitors. These event monitors communicate with FANUC Robotics robot server program that handles robot to computer communication. An event monitor can be issued to monitor specified data for changes. The occurring data changes launch a new VI that contains wanted actions. These user panel values do not have to be updated in every program cycle, but only when a change is detected. Using event monitors makes the program more efficient by saving computing time for other processes.

Monitoring is handled with Reg Event Callback LabVIEW node. It defines what value is monitored and what VI is run when an event is detected. Event Callback VI acts like a subVI.

A good example of an event is alarm update of MATS user panel. When the robot server detects a new alarm in the robot controller, it raises an event. This event is detected by Reg Event Callback node that is connected to the alarm's property of IRobot object. The detected event launches AlarmNotify Event Callback subroutine that updates alarms from controller to user panel.

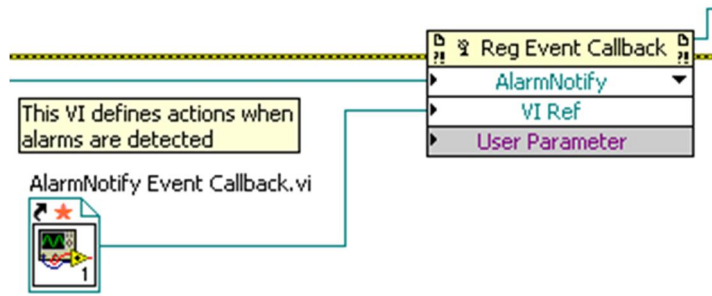


FIGURE 24. Example usage of Reg Event callback.

Event monitors are started at the beginning of a robot program. This is done by placing the Start MATS user panel LabVIEW program at the beginning of every test program. Start MATS user panel LabVIEW program is run simultaneously with the main LabVIEW program. It contains all event monitors and other user panel objects that have to be updated during running program.

The user panel does not have any security related actions like emergency stop. Ethernet connection is not a traditional automation field bus, and it cannot be trusted to perform this critical action. Additionally, the user panel values are not updated regularly so it would be dangerous to place critical controls to this virtual user panel.

The objects of MATS user panel are logically arranged in groups. Values that are somehow related to each other are placed inside a box. This way the operators can have an intuitive understanding of the user panel. More vital information is placed to the left side of the user panel. This arrangement mimics other computer programs. For example in Windows XP Start menu and desktop the icons are placed on the left side by default.

3.7 Tool identification

It is important that the right tool is used for right test sequences. MATS was built in a way that it would be capable to identify the tool that the robot is trying to pick up from the tool stand. Tool identification ensures that correct tool is used in all test sequences. Five of the twelve signal lines of SMC AHC system are used for tool identification. These signal wires are connected to FANUC

Robotics digital input modules. Robot tools are built in a way that every tool has a unique binary identification number. Identification numbers are formed by connecting proper digital input wires to digital output wire. This gives possibility to identify 14 different tools.

Tool identification subVI detects if the robot is picking up a desired tool. Tool number must be specified for tool identification subVI. This is done only once during the test creation. Tool identification subVI gives a Boolean value as an output. If the correct tool is picked up the output is true. Boolean output can be used as an input value of LabVIEW case structures. With LabVIEW case structure it is possible to react differently in cases where the wrong tool is detected.

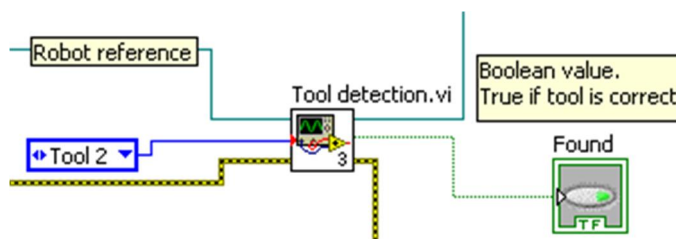


FIGURE 25. Usage of the tool identification subVI.

Tool changing subVI has an inbuilt capability to identify the tool, and search the correct tool from all of the tool stands. If the robot does not find the correct tool from any of the tool stands, it will notify the operator with an email. The positions of the tool stands are taught in advance so the robot programmer does not need to re-teach them at any point.

3.8 Test project handling

LabVIEW has an inbuilt tool to manage LabVIEW projects. This project tool is called Project Explorer. It can be used to create and edit project files. With Project explorer it is easy to group LabVIEW and non LabVIEW files to groups, manage dependencies and resolve conflicts. Project Explorer folders can be defined to be auto-populating. This way files that are added to auto-populated folders are automatically added as part of the project. (Using Lab-

VIEW Projects 2007).

LabVIEW project explorer's ability to manage all file types makes it an efficient tool to manage all test related data that MATS provides. It can be used to manage LabVIEW subroutines, position files, test instructions and test results. This way there is no real need to have different project and data management software when using MATS. This adds a great deal of usability to the system.

With LabVIEW project explorer test developers can manage all of the test related files. Project manager keeps automatically track of new position files and subVIs.

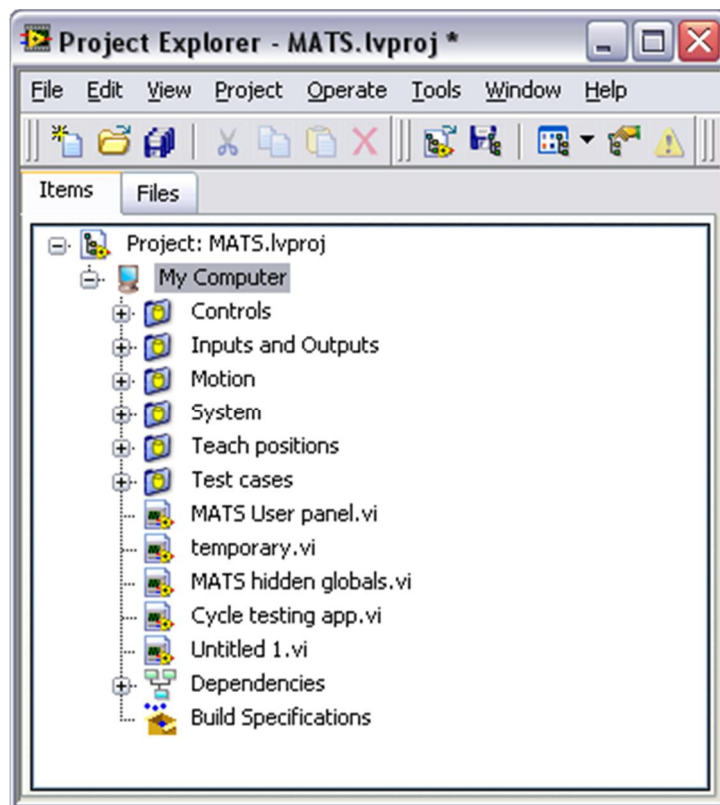


FIGURE 26. LabVIEW Project Explorer window.

One down side of LabVIEW project explorer is that with it the project files have to be managed locally. There is no possibility to publish or manage test files online in another computer. On the other hand this makes the system more secure. Ability to publish and manage sensitive data through network might be considered a security risk.

3.9 Ethernet security

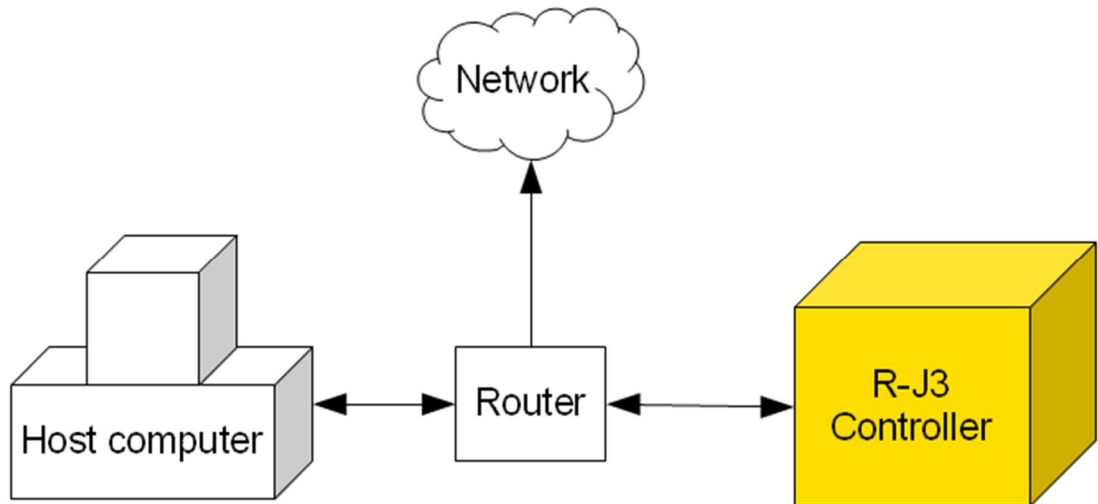


FIGURE 27. Network connection

R-J3 controller has an access control IP list that defines computers that can access the robot controller. The computer's IP address must be in this IP list, otherwise the controller will deny all connection attempts. R-J3 controller has feature where users could be identified with username and password. In this case user name and password access control is not being used because user name and password exchanges might increase the controller's response time.

R-J3 controller is connected to the company network, but the router is configured in a way that only the host computer that is in the same room with MATS can access the R-J3 controller.

Cassidian Finlands company network is restricted and can be considered as a safe environment. The room where the robot is located is locked at all times. Only few company employees have access to this room. Strict physical access control increases the security of the system.

4 EXAMPLE CASE

4.1 General

This section tries to open the overall process of test creation with Cassidian Finlands MATS. The values given in this example test case do not represent any actual mechanical test. The values have been changed due to confidentiality issues.

4.2 Test Design

If there is a new product or new part to be tested, the first part of the test creation process would be to design a robot tool and attachment for the product. Robot tool design and manufacturing are time consuming processes. In between the design phase and Position teaching phase there can be several weeks in between. This extra time comes mainly from designing and ordering robot tools and attachments. If there is no need to order new accessories, test the creation process is much faster. A functional test can be created in less than eight hours.

As the test definition illustrates, a tool that is used in this test case has to be able to push THR9 power and volume buttons. For this example test it is assumed that these tasks can be performed with the same robot tool. It is also assumed that the robot tool is ready, and can be used immediately.



FIGURE 28. Collection of Cassidia Finland's TETRA products

The test is documented in a way that it can be repeated every time when needed. The test object positions, position files and robot program files are saved so the test can be quickly run when needed.

In the design phase a first draft of robot movements are created. When creating motion sequence drafts for test programs, it was a good practice to first create a motion path sequence table. Motion sequence table could then be used during position teaching as a template. Motion path sequence should contain at least position index, motion type and comment. Some transitional positions might have to be added during position teaching.

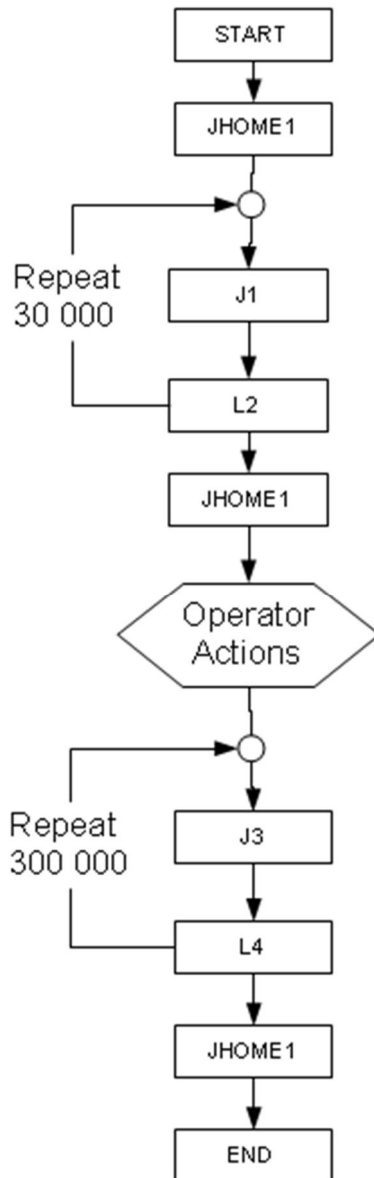


FIGURE 29. Position sequence flow chart of example test case

In this example case only four positions have to be taught. The first position is above the power key, and the second one is when the power key is pushed down. After THR9 is turned sideways the third position is above volume up key and in the fourth position the volume up key is pushed down. There may be need to train additional transitional positions, but there are also pre-taught safe positions inside a work area. Transitions between safe positions could be performed with joint motion type. This motion type is faster, but it will not necessarily follow a straight path. The actual pushing of a key is done with the linear motion type. The linear motion is not as fast as the joint motion, but it

travels a straight line between the positions. The position sequence for these coordinates would be as follows.

Position sequence indicates the type of the motion, index of position and order of motions and actions. For example J1 means that robot moves to first position of the position file with Joint motion type. L4 means that robot moves to fourth position in position file with Linear motion type.

HOME1 position is the pre-defined safe position above workbench 1. After position sequence has been created, the first draft of robot program can be created.

4.3 Definition

Normally test definition is based on a test standard or detected fault that has to be verified or examined. In this example test case the values and test case itself are invented. Even if this use case is a fake, it will give a good overview about how MATS can be used.

The first step of test creation process is filling in a test specification template. It provides information about the test for test designers and testers. It specifies the needed hardware components, preconditions, robots actions and post-conditions of the test. At this point the template is only a draft, and it can be updated when there is more information about the test.

TABLE 1. Example test specification template.

Test 1988	
Test ID	1988
Status	Draft
Date/version	14.01.2010
Name	THR-9 keys and keyboards

Description	
Purpose of test	Keys and keyboards mechanical reliability. 9 samples required.
- Configuration	Fully assembled THR-9 radios. 3 radios on table each time. Test repeated three times to collect data on 9 samples.
- Test options	Number of radios to be tested in this run. (Default is 3 radios)
- Reference documents	
- Supported HW version	Standard THR-9 radios.
- Test duration	Approx. 3 hours for one radio.
Pre-condition	Place product specific tool head "Finger tool" in tool post no 1. Place 1 to 5 "THR – connector test" jigs to test table A. Coordinates "B-5, B-10 and B-15". Insert 1 to 3 radios on jigs screens upwards. Test buttons functionality.
Test steps	STEP 1: Push power key 30.000
	STEP 3: E-mail notification to change radio position
	STEP 4: Tester positions all radios volume buttons upwards
	STEP 5: Start next phase from robot monitor centre
	STEP 6: Push volume up key 500.000 times
	STEP 7: Email notification about test completion
Criteria	Characteristics must be within specified limits. Key should give contact every time. Contact must happen immediately after tactile point without any extra force.
Test data	Pass/fail/inconclusive
Post-condition	Remove the product specific jigs. Functionality of the keys must be checked at least at temperatures of -35 and +60 degrees. Functionality of test buttons.

In this example test Cassidian THR9 Professional Mobile Radio **power key** and **volume up** button's endurance was tested. The tool used in this example is mechanically designed in a way that it presses the keys with at desired force. The **power-key** is pressed 30 000 times. Then device under test is

turned by an operator and **volume up** key is pressed 300 000 times. Values or test progression do not represent any actual test.

4.4 Teach positions

Position teaching is handled with Teach positions LabVIEW program and R-J3 Robot controller's teaching pendant. With the teaching pendant the robot is jogged inside work area. When the desired position is reached, the position is saved to the position file by clicking a button in Teach positions program's front panel. The position teaching for these four positions would take about fifteen minutes if the system start up time and position sequence testing were included.



FIGURE 30. FANUC Robotics Teaching pendant.

When pushing a key of a device under test, the key contact can be verified either with an installed force transducer or by wiring key contacts inside the THR9 to outside of the hull. By connecting these contact lines to the test signal box, a LabVIEW function could verify a successful key contact.

Teach positions LabVIEW program has an option to drive the robot between positions with the wanted motion type. This way the test designer can immedi-

ately test and retouch the positions that have been created.

4.5 Program creation

A first draft of position teaching has been created in a design phase of the test creation process. Before starting to create a robot program from the test, it is recommended to perform a step test from the taught positions according to position sequence table. This way it can be ensured that the taught positions are correct, and no transitional positions are needed. Positions can be added to position files at any time with Teach positions utility. Example test program is represented as a whole in appendix 4.

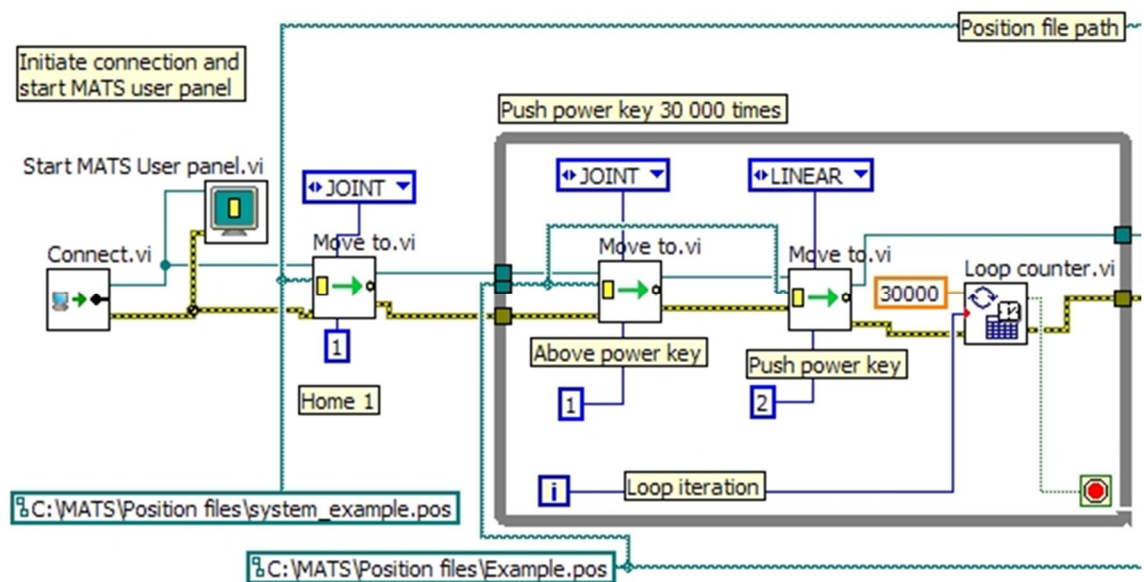


FIGURE 31. First part of the example robot program.

The first part of this robot program is a “Connect” subVI. It initiates the robot connection, so that other methods can use the robot object. The second routine is the starting of MATS user panel.” Start MATS User panel” subVI starts event monitors and updates the MATS user panel.

The first Move to subVI drives the robot to a safe position above the workbench 1. This movement is added because it is not sure where the robot is inside a work area.

Motions that push the power key are placed inside a while loop structure. This way the push motion can easily be repeated for 30.000 times. The push motion is formed from two positions. The first one is located above the power key, and the second is the position where power key is pushed down. When the robot moves between these positions it creates the push motion

When the “Loop counter” subVI is given the wanted cycle amount and the loop iteration as an input, it gives a Boolean output value that can be used to stop the loop when desired amount of cycles have been performed. The “Loop counter” also calculates the estimate how long this particular loop runs, and updates the time information to user panel.

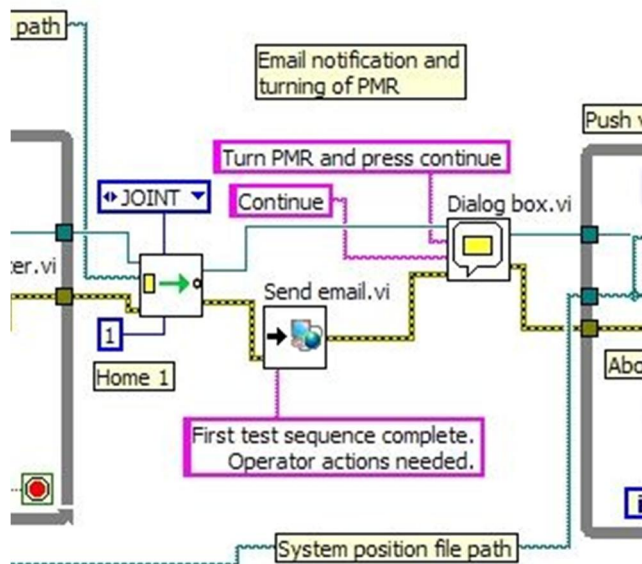


FIGURE 32. Second part of the example robot program.

When the loop iteration is 30 000 the stop condition is set to true. After this the robot is driven to a safe position above the workbench 1. “Send email” subVI sends an email notification for the test operators. Email receivers are determined in MATS user panel. “Dialog box” subVI stops the program execution. It reserves the robot reference and lets the operator to change the position of the test object. Program execution continues when the operator pushes button from dialogue box that pop ups when program execution reaches the “Dialog box” subVI.

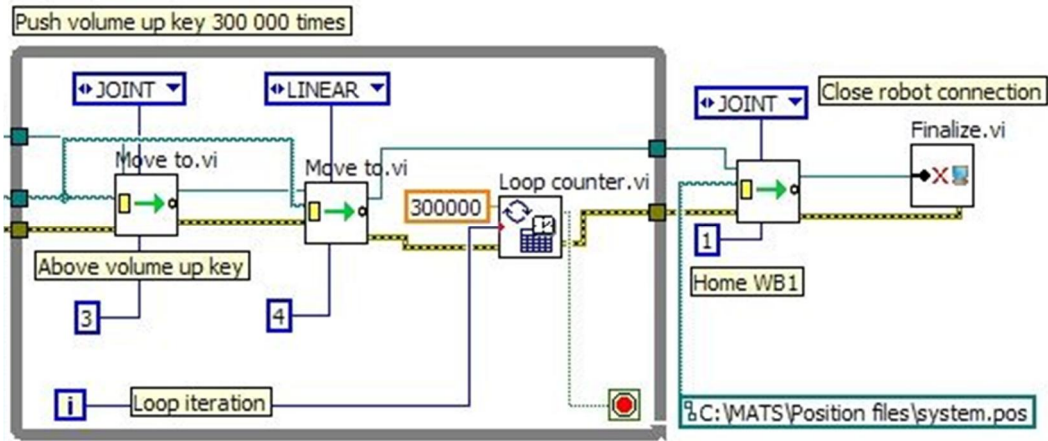


FIGURE 33. Last part of example robot program.

This section of the robot program is copied from the first loop. Copying similar code segments with LabVIEW is easy and saves time. Pushing operation the same as in the first push operation. This time volume up key is pushed for 300 000 times.

The last Move to subVI drives the robot to a safe position above workbench 1. “Finalize” subVI closes the robot connection and event monitors. It also sends a notification email for the operators. After the test program is created, it can be tested with low speed. The speed can be controlled with the teaching pendant. Testing of the robot program ensures that the robot performs all needed operations correctly.

5 RESULTS

An outcome of this thesis is a working test creation environment that utilizes an industrial robot. The environment can be easily modified if new hardware components are introduced. The test creation environment is also easy to use as can be seen from MATS usability study from appendix 5.

The largest part of the work hours put to this project has been spent to connecting FANUC Robotics R-J3 robot controller with LabVIEW. Trying different connection methods and interfaces before choosing the one described in this

thesis was very time consuming.

Another large time consumer was the creation of the LabVIEW subVIs in a way that the test programs would be easy to create. Figuring out the internal structure of PCDKs ActiveX components was the main duty in this part of the project. Correcting detected structural faults from LabVIEW code took equally much time. This time was largely lengthened by the fact that there were no earlier studies or examples published about this subject. Additionally inadequate PCDK documentation caused few bad surprises and even development halts.

The creation of LabVIEW programs for position teaching and robot monitoring was an easy task after the structure of the ActiveX components was resolved. Some consideration was put into usability and alteration of the user interfaces of these programs after user experiences had been received.

Integrating LabJack U6 data acquisition card to the system was straightforward task because LabJack provided compatible LabVIEW functions in the package. It only took a small time to convert these functions to look like subVIs that I had created for the robot. This way test designer will not get confused about the different structure of the data acquisition cards LabVIEW subVIs.

The last part of the project was the documentation. This documentation on my part included writing the software and hardware user manuals, LabVIEW subVI list with functional descriptions and an annual maintenance table.

6 SYSTEMS FUNCTIONALITY AND USABILITY

The functionality of the test creation systems software was verified by creating and running test cases that resembled actual tests. The tests cases were simplified in a way that robot did not affect any actual objects. This way it could be assured that occurring faults were caused by the system's software and not by

the hardware parts.

The test system was left to perform a test for a weekend. On Monday it was verified that the test was still running. This ensured that there were not any software crashes or malfunctions during the weekend test session. Some of the test runs were interrupted by system crash, but crashes were caused by faults in the test program design, not in the system software itself.

LabVIEW's and robot server's memory consumptions were written down in the first cycle of the weekend test. These figures were then re-checked after the weekend, and verified that they were not above the normal levels. This test ensured that LabVIEW functions and subVIs did not leave any unclosed objects in the computer's memory.

The overall usability was verified by interviewing the person who uses this system. A form was created that consisted of questions about the usability of the system. The answers of this questionnaire can be seen in the appendix 5. These results can not be used widely because only one person has taken the test. However the usability of the system can be verified from the results.

There is no real need to create more accurate tests about the software's functionality at this point of the system development. For the employer, it was enough that the system works as requested.

7 REQUIREMENTS

This thesis project was a very challenging task for me. At the beginning everything looked fairly simple. The first time estimation for the completion of the project was three months. In the end this project took about a year. A major part of it was part time work, done during studies. People at the Cassidian Finland were flexible and gave the author more time and resources. Thanks to them, the end product of this thesis is a functional and an easy to use mechanics automated test system.

I had some robot studies in the university before this project, but they consisted mainly of robot programming and light peripheral communication exercises. The university had no Fanuc robots available for learning. For this thesis I had to examine and learn the functionality of the R-J3 controller very deeply, much deeper than I even knew was possible from my classes. I took part in the mechanical maintenance and troubleshooting operations that had to be done for robot before it was functional.

I also learned the TP language that FANUC normally uses for robot programming. I used this language early in the project to test the robot's functionality and capabilities of the peripheral devices. I created robot programs with TP language with FANUC's offline programming applications and Teaching Pendant. I also learned a bit of VisualBasic programming language while interpreting FANUC's PCDK program examples.

One significant thing that I learned during my thesis project was the knowledge acquisition from different sources, and making the decisions and interpretations early on based on this knowledge. After the first two months working in this project I understood that there were no persons or documents available who could explain how get LabVIEW and R-J3 controller to communicate. People at the Cassidian Finland did not know that much about robotics, and people at the Finland's FANUC importer Fastems did not know much about LabVIEW robot applications. Combining the knowledge from all the sources and applying it to this project was a skill that I learned.

Lack of publications about this subject added a great number of difficulties to the project. There were only some different forum discussions where was mentioned that someone somewhere in the world had controlled a FANUC robot with LabVIEW. Everything else was up to me to learn and to discover.

The referenced actions of learning and discovering are actually a small understatement when talking about the knowledge that I absorbed from PCDK's and LabVIEW's combined functionality. It was not enough to find out material and then apply it to the system. I had to really internalize the information about the

PCDK's structure, and then use this information in a way that FANUC, National Instrument nor any other source have documented, or at least published.

The useful issue in my education was the LabVIEW experience I had from many of the courses. We never actually handled ActiveX components with LabVIEW but still the existing LabVIEW skills helped. Additionally the idea of a robot being just another programmable machine came from university studies. During school exercises I discovered that there was nothing difficult about robots. It is basically just a computer with an arm. This mentality helped a great deal during this project.

8 CONCLUSIONS

8.1 General

The project that this thesis describes was quite different from “traditional” robot applications. Usually industrial robots perform particular programmed task for long periods of time without any need of reprogramming. They may react to sensor and vision information; however the same program might be used for several years without any changes.

In this project the goal was almost the opposite of traditional robot programming. A whole test creation system had to be created that would be modular and highly adaptable because one test case might only be performed once. Every new fault or mechanics detail would need a new test program to be created.

8.2 Future improvements

One problem that occurred when running the robot programs from the host-computer was that a robot paused for a short time between motions. The length of the pauses was about a tenth of a second, but when the robot performed tens of thousands of cycles, these pauses increased the overall of the

test time considerably. Eliminating the pauses would allow tests to be run during one work day.

When running normal TP programs from the robot controller's memory, these pauses did not exist. The reason for these pauses must be in The Ethernet connection or in LabVIEW's habit of running programs slower than "traditional" programming environments.

The Data transfer between the robot controller and the host computer should be minimized. This might make pauses shorter. Also R-J3 controller's Ethernet card is working in half-duplex speed. FANUC Robotics might have replacement Ethernet card that works in full-duplex mode. This might reduce the data transfer time, and the length of the pauses.

The system's error handling also has some issues that could be improved. The test programs work perfectly when the test designer follows the instructions. If the test designer makes a programming mistake and runs the faulty program, the whole environment might crash. The tests are easy to start all over again, however this is not a desired feature. The system should have some kind of error recovery method, or subVIs could be created in a way that the test programs will not run if they do not have all the correct data in all inputs.

The last and minor improvement would be to add MATS LabVIEW subVIs to LabVIEW functions menu under their own segment. This way the test creation would be even easier.

8.3 Achieved goals

The main goal of the thesis was to create an environment that could be used to test the mechanical endurance and wear resistance of professional mobile radios, their parts and the peripherals. This goal can be considered to be achieved because some tests have already been run successfully. According to the person who created the tests the system gives useful data that can be

used in professional mobile radio development.

An other significant goal was that the usability of the system should be high. MATS usability study shows that this goal has also been achieved. Test creation is carried out with a graphical program creation environment and there is no need for line based programming. Functional tests can be created in a few hours. Also the usage of LabVIEW for test program creation, position teaching, robot monitoring and project management makes test creation and running simple. All peripherals are also connected to LabVIEW.

9 SOURCES

Brooks, R. 2002. Robot: The Future of Flesh and Machines. London: Penguin Books.

Creating SubVIs. 2008. National instruments support website. Referred 28.2.2011. http://zone.ni.com/reference/en-XX/help/371361E-01/lvconcepts/creating_subvis.

EADS Defence & Security changes its name to CASSIDIAN. EADS Global website. Referred 27.2.2011. <http://www.eads.com/eads/int/en/news>, Press releases, EADS Defence & Security changes its name to CASSIDIAN.

Jalovaara, V-M. 2010. Onko robotista rajavartijaksi? Tekniikka & Talous 08.10.2010, 6.

Robotiikka. 1999. Toim. R. Kuivanen. Vantaa: Suomen Robotiikkayhdistys Ry.

Talouselämä 500. 2011. Talouselämä 500 website. Referred 9.3.2011. <http://www.talouselama.fi/te500/?view=company&cid=70587>.

Travis, J. & Kring, J. 2006. LabVIEW for everyone: Graphical Programming Made Easy and Fun -- 3rd ed. Crawfordsville: Pearson Education Inc.

Using ActiveX with LabVIEW. 2006. National Instruments support website. Referred 1.10.2010. http://zone.ni.com/reference/en-XX/help/371361B-01/lvconcepts/using_activex_with_labview.

Using LabVIEW Projects. 2007. National instruments support website. Referred 28.2.2011. http://zone.ni.com/reference/en-XX/help/371361D-01/lvconcepts/using_labview_projects.

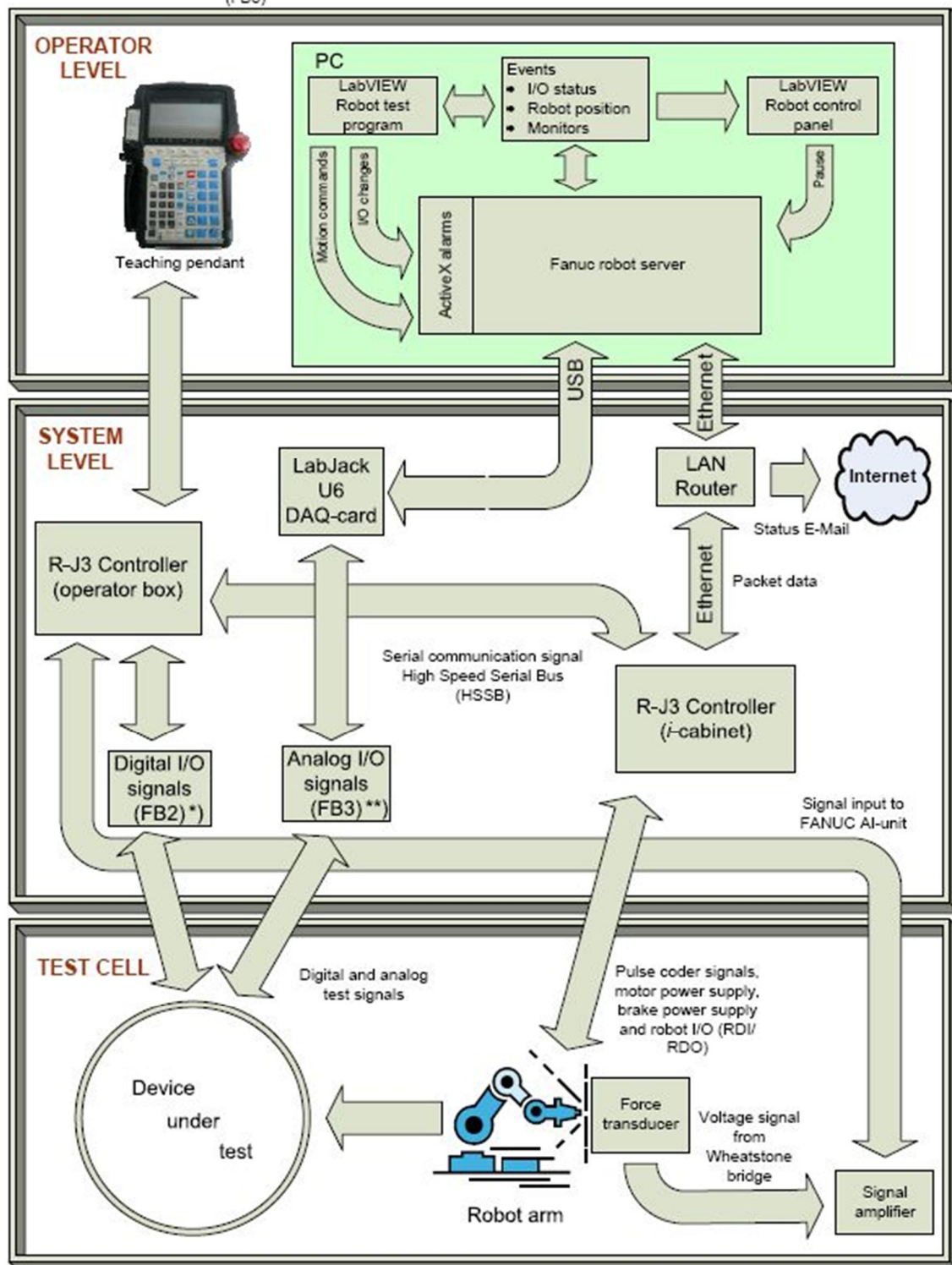
10 APPENDIX

APPENDIX 1. MATS Block diagram

*) User operable interface for additional I/O signals (FB2)
 **) User operable interface for analog measurement signals (FB3)

System block diagram

Monday, June 28, 2010



APPENDIX 2. LR Mate 200i data sheet

LR Mate 200i

Basic Description

The LR Mate 200i is a six-axis, modular construction, electric servo-driven, tabletop robot designed for a variety of manufacturing and system processes. The FANUC LR Mate 200i robot is engineered to provide maximum flexibility in a compact design and is supported by our extensive service and parts network.

LR Mate 200i, the Solution for:

- Material handling
- Parts transfer
- Assembly
- Material removal
- Dispensing

Benefits

- Tabletop size allows use in a wide range of applications from commercial to industrial installations
- Upright or invert mounting positions with no robot modifications
- Small footprint allows placement in tight surroundings
- Hollow joint construction encloses all cable routing to eliminate snagging
- Absolute serial encoders eliminate the need for calibration at power-up

Features

- 6 axes of motion
- 3kg (6.6 lbs) payload capacity
- +/- .04mm (+/- 0.002") repeatability
- Axes speeds of up to 480° per second



- End effector connector built into the wrist
- Integral, internally mounted solenoid valve pack
- Fail-safe brakes on axes 2 and 3
- R-J2 Mate Controller features low voltage I/O: 20 inputs (8 dedicated), 16 outputs (4 dedicated) and 4 inputs at end-of-arm connector
- Optional R-J3 Controller supports FANUC Robotics KAREL application software and standard FANUC Robotics I/O modules
- All motors, cables and tooling services are located internally for protection from environment
- Standard IP54 dust and liquid intrusion package

Options

- Available with R-J2 Mate Controller for basic material handling or R-J3 Controller for complex, process applications
- FANUC Robotics SPI Interface package (hardware and software) to connect robot to injection molding machines
- FANUC Sensor Interface serial communications software allows robot to exchange data with third party devices such as bar code readers, vision systems and PCs
- Data Transfer Function serial communications software allows robot two-way data exchange with a PC; users can control robot actions through VB graphical interface

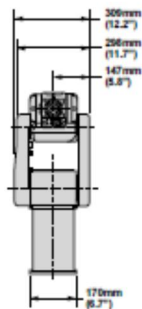
Reliability Features

- Based on a proven, reliable design of the FANUC LR Mate 100i, which has an installed base exceeding 4,000 units worldwide
- Brushless AC servo motors minimize motor maintenance
- Harmonic drives on all axes result in higher reliability and reduced maintenance costs
- Sealed bearings and drives

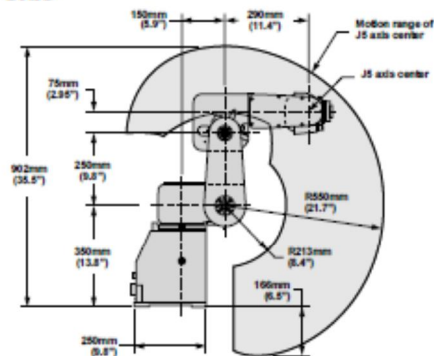
FANUC
Robotics

LR Mate 200i Dimensions

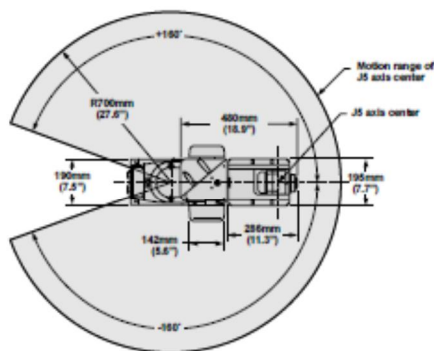
Front



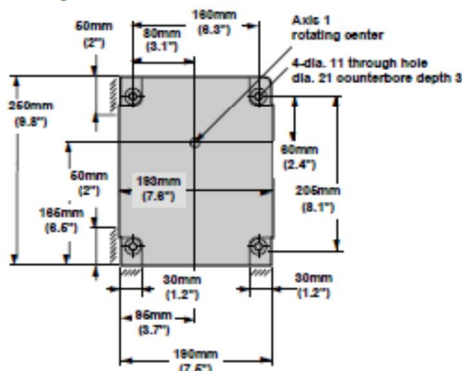
Side



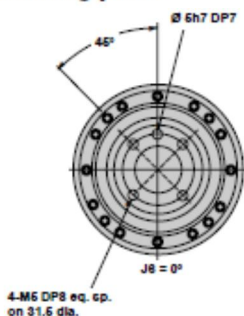
Plan



Footprint



Mounting plate



LR Mate 200i Specifications

Items	E	
Axlec	3kg (6.6 lbs)	
Payload	Range	
Motion range and joint speed	J1	320° (± 160°)
	J2	185° (+ 152°, -32°)
Moment	J3	365° (± 138°)
	J4	380° (± 190°)
Inertia	J5	240° (± 120°)
	J6	720° (± 360°)
Mechanical weight	J4	55.5 kgf • cm
	J5	55.5 kgf • cm
	J6	40.0 kgf • cm
Controller	J4	1.1 kgf • cm • s ²
	J5	1.1 kgf • cm • s ²
	J6	0.41 kgf • cm • s ²
Mechanical weight	39kg (85.8 lbs)	
	Controller R-J2 Mate or R-J3	
Reach	700mm	
Repeatability	±0.04mm (±0.002") based on JISB8432	
Mounting method	Upright/inverted	
Mechanical brakes	Axis 2 and Axis 3 (Axis 1 option)	
Dust/water intrusion protection	Conforms to the IP54 standard for dust and liquid protection (seals may need periodic replacement if used with chlorine or gasoline based coolants)	



A New Age of Industry

FANUC Robotics North America
 3900 W. Hamlin Road
 Rochester Hills, MI 48309-3253
 Phone (248) 377-7000
 Fax (248) 276-4133

Literature Request
 1-800-47-ROBOT

Charlotte, NC
 Phone (704) 596-5121

Chicago, IL
 Phone (847) 898-6000

Cincinnati, OH
 Phone (513) 754-2400

Los Angeles, CA
 Phone (949) 595-2700

Toledo, OH
 Phone (419) 866-0788

Toronto, Canada
 Phone (905) 812-2300

Montréal, Québec
 Phone (450) 492-9001

Mexico City, Mexico
 Phone 52 (5) 611-5998

Aguascalientes, Mexico
 Phone 52 (49) 10-8000

Sao Paulo, Brazil
 Phone (55)(11) 3955-0599

web page: www.fanucrobotics.com

©1999 FANUC Robotics North America, Inc. All rights reserved.
 FRNA-10/9-06-005

FANUC ROBOTICS LITHO IN U.S.A.

APPENDIX 3. R-J3 Data sheet

SYSTEM R-J3 Controller™

Basic Description

FANUC Robotics' SYSTEM R-J3 Controller uses advanced technology packaged in a proven, reliable third generation controller design. Process capability and open architecture features improve application and motion performance while simplifying system integration.

SYSTEM R-J3 incorporates FANUC Robotics' unique "plug-in options" concept, which allows flexibility for application specific configurations while maintaining a commonality for all users of the system.

Hardware Features/Benefits

- Use of surface mounting and 3-D packaging reduces components and increases reliability.
- 32-bit main CPU with dual processor architecture permits fast calculations, reduces program execution times and increases path accuracy.
- Quick change servo amplifier reduces maintenance time.
- Distributed I/O options reduce cabling costs and simplify troubleshooting.
- Provides extensive line of compact I/O modules for both digital and analog signals.
- Allows for fast power-up and program execution with auto resume after cycle start.
- Increased use of fiber optics simplifies connections and enables faster communications.



System Features/Benefits

- Ergonomically designed, lightweight teach pendant with large, easy-to-read backlit LCD display
- High-speed, precision control of up to 16 axes of motion
- Auxiliary axes options can support up to three separate motion groups, each with its own control program and simple kinematic models.
- Multi-tasking operating system allows execution of several concurrent user programs.
- Advanced communications and networking capabilities include built-in Ethernet and PCMCIA interface.

Process Features/Benefits

- AccuPath provides enhanced path tracking during linear and circular motion while minimizing speed variations.
- Instant trigger response (<4ms) increases repeatability and improves process control.
- Collision detection minimizes potential damage to the robot or end-of-arm tooling.
- Zone I/O provides application flexibility by monitoring and controlling robot interface signals independent of the taught path.
- Coordinated motion simplifies the teaching of part programs on a moving table or positioner.
- TurboMove provides minimal cycle time by computing robot dynamics in real-time.

FANUC
Robotics

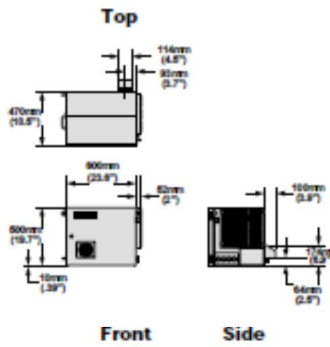
R-J3 Standard Configuration

Items	Specifications
i-size Cabinet (Integrated or remote)	See drawing for dimensions
Operating environment	- Ambient temperature: 0-45°C - Humidity: 75% RH or less non-condensing (95% max) - Vibration: 0.5G or less
Power supply	Three phase 200-575 VAC +10%, -15%, 50/60Hz ±1Hz with circuit breaker
CPU	32-bit dual processor architecture (separate motion and communication) with real-time clock/calendar
Controlled axes	16 (up to three motion groups)
Serial/host communications	- Built-in Ethernet (10BaseT) - FTP: allows simple file transfers to a variety of host platforms - Ethernet Controller Backup/Restore: provides backup and/or restore of the robot Controller memory image - Three RS-232 ports (one can be configured as RS-422)
Teach Pendant	Back-R LCD, multi-function

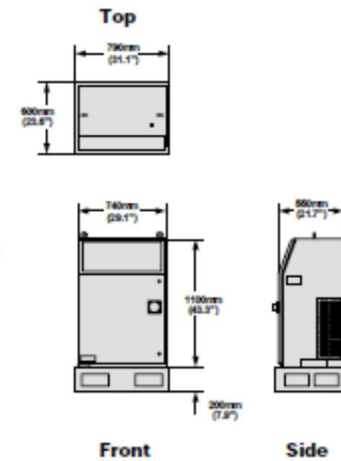
R-J3 Options

Items	Specifications
B-size Cabinet	See drawing for dimensions
IO sub-systems	- Model A (modular rack mounted - 5 or 10 slots) - Model B (distributed DIN rail mounted)
IO types	- DUDO: 512 point maximum each (Includes process I/O) - Digital AC or DC input modules - Digital AC or DC output modules - 12-bit Analog input or output modules
Process I/O	- Digital input: 40 points maximum - Digital output: 40 points maximum - Multiple points can be utilized as a code (group I/O) - Analog inputs: 8 points - Analog outputs: 2 points - Digital input for welding: 8 points - Digital output for welding: 8 points - Wire stick detect
Remote I/O sub-systems	- Allen Bradley Remote I/O interface - Devicenet (1 channel in i-cabinet, up to 4 in B-cabinet) - Profibus DP slave
Host communications (Ethernet-based)	- ALU (hardware interface) - PC Interface: enables PC application communication
Diskette drive	- 3.5" HD MS-DOS format (PS-110) - IBM-PC compatible disk emulator program
Memory card for system software installation or program backups	PCMCIA type 2 interface for: - 2MB SRAM card - ATA flash disk cards (SanDisk compatible)

i-size Cabinet

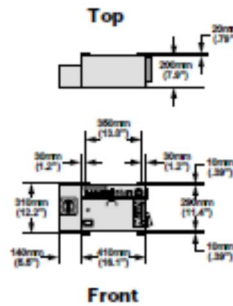


Optional B-size Cabinet

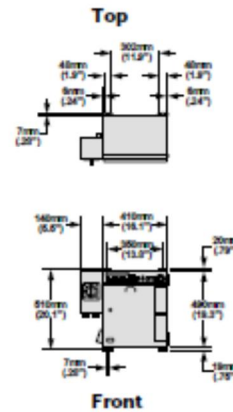


OP Box Configurations

Type A Op Box



Type B Op Box



A New Age of Industry

FANUC Robotics North America
3900 W. Hamlin Road
Rochester Hills, MI 48309-3253
Phone (248) 377-7000
Fax (248) 276-4133

Literature Request
1-800-47-ROBOT

Charlotte, NC
Phone (704) 596-5121

Chicago, IL
Phone (847) 898-6000

Cincinnati, OH
Phone (513) 754-2400

Los Angeles, CA
Phone (549) 595-2700

Toledo, OH
Phone (419) 866-0788

Toronto, Canada
Phone (905) 812-2300

Montréal, Québec
Phone (450) 492-9001

Mexico City, Mexico
Phone 52 (5) 611-5998

Aguaascalientes, Mexico
Phone 52 (49) 10-8000

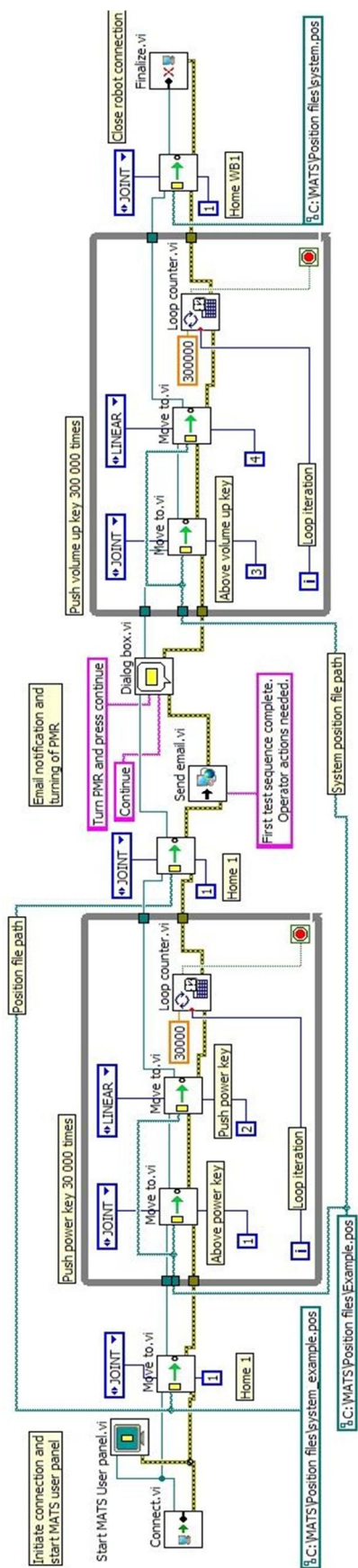
Sao Paulo, Brazil
Phone (55)(11) 3955-0599

www.fanucrobotics.com

©1999 FANUC Robotics North America, Inc. All rights reserved.
FRNA-99-05-013

FANUC ROBOTICS LITHO IN U.S.A.

APPENDIX 4. Example test program



APPENDIX 5. MATS Usability study

MATS usability study						
This inquiry is designed to verify usability of Mechanics Automated Test System (MATS). Test has been performed for one person who has created and performed mechanical tests with MATS						
Personal information						
Job title: HW Test Engineer						
General						
		1	2	3	4	5
How experienced are you with robot related tasks before using MATS. Robot related task are eg. Position teaching, robot programming, robot simulation.	I have never worked with robots	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
						I work with robots every day
What is the level of your LabVIEW experience before using MATS	I have never used it	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
						I use LabVIEW every day
What is the level of your programming experience before using MATS	I have never programmed	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
						I program every day
How well you have read the hardware documentation of MATS	Not at all	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
						Very well
How well you have read software documentation of MATS	Not at all	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
						Very well
How many hours have worked with MATS (by estimation)		100 : hours				
Will this system be used to create mechanical tests in your company	No	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
						Certainly
Software usability						
		1	2	3	4	5
How easy it is to teach positions to robot with MATS.	Very hard	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
						Very easy
How easy it is to create functional robot programs with MATS.	Very hard	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
						Very easy
What school grade would you give for MATS's software implementation	Passable	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
						Excellent
How many grades does remaining software bugs affect the outcome of previous question	1 grade	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
						5 grades
How many lines of programming code did you write when you created tests with MATS (by estimation)		0 : lines				
How long it took in average to create functional		7,5 : hours				

APPENDIX 6. Teach positions LabVIEW program



APPENDIX 7. MATS user panel

