

Loader Cart Automation Enhancement

Wei Hoo Chong*

*PCR Software Department, Motorola Solutions Malaysia Sdn. Bhd., Penang, Malaysia.

hoochong@motorolasolutions.com

Abstract—Target market of PCR¹ 2-way radio is people with critical mission. Thus, quality of radio has to be ensured. Loader cart system is being used actively as an automated testing tool in PCR during system test phase. It is able to stress the radio to its limit to ensure its stability, robustness, and reliability. All major issues reported have to be resolved before end products are shipping to customers. Loader Cart System is used to reproduce complicated system issue repeatedly that was triggered by the same system because it might be time consuming for human to reproduce the same² issue manually. In order to do that, the loader cart system has to be configured before it can operate. However, there is a lot of appraisal works and time needed in configuring the same testing environment of Loader Cart System for issue reproduction. This process takes the longest time from developers and it is tedious to be carried out manually. From experience, the environment setup on 8 radios (a set of loader cart system) took ~4 hours. This amount of time is just for one cycle and the time taken for debugs, implementation, and firmware compilation has yet been considered. Hence, this paper aims to reduce “non-value-added” appraisal works of developers and improve loader cart automation test cycle time. The proposed solution is automating this “non-value-added” appraisal works by using free automation tool such as AutoIT that required no expensive license. The proposed solution is estimated to significantly improve the efficiency by at least 85%, which reduces the time needed from ~4 hours to ~30 minutes.

Keywords— Loader Cart System, automation, efficiency, reduce, appraisal works, cycle time improvement, AutoIT

I. INTRODUCTION AND PROBLEM STATEMENT

Software testing is an important software engineering activity widely used to find defects in programs. The purposes of testing can be for quality assurance, verification and validation, or reliability estimation. As we know, there is no perfect software in the world. However, when there is an issue reported by test team, as a developer, we need to understand the issue thoroughly before starting to fix it or root cause it. This includes knowing the testing environment and how to reproduce the issue. This is to ensure that we are fixing the “correct” issue efficiently. However, some issues may need a

longer time to surface (hardly reproduce), for example, the wrong option³ or wrong message received or sent from or to other components in particular or special situation that may cause the system to hang. We classified this as system issue. The occurrence rate is may be 1/1000. Hence, the automation tool is used to reproduce this kind of issue. There is a bunch of test tools and systems available in this world for example Hopper[2] in mobile testing environment. However, in this paper, we focus on Loader Cart System. Loader Cart System is actively used during System Test phase. The purpose is to stress the radios to its limit to test the radio’s stability, robustness, and reliability. This process ensures the company is producing high quality product or system to the end user. Basically, if a complicated system issue had been reported and was triggered by the Loader Cart System, developer will try to reproduce the issue with the same system as well. This is to ensure same test environment and same test procedures been used in issue reproduction. Loader Cart System is not only used in issue reproduction but also used in issue verification and validation as well as when the issue is been resolved. However, too much appraisal works need to be done in configuring the same testing environment of Loader Cart System repeatedly. This process takes the longest time from developers and it is tedious to be carried out manually. Statistically, the environment setup for a set of loader cart system, which contains 8 radios, took ~4 hours in average per cycle from developer. Kindly refer to Manual Flashing performance line in Figure 6 below for details. Hence, in this paper, we propose to automate Loader Cart System configuration works to reduce at least 85% of “non-

¹ Professional Consumer Radio

² Same test environment and test steps.

³ There are many options used in Radio Firmware implementation, such as Mic Option, Speaker Option, etc.

value-added” appraisal works of developers and improving testing cycle time by 2.7x or ~63% of each cycle. The solution is simplifying the existing manual process to become more cost effective way while remaining the existing apps or tools that are currently used.

The rest of the paper is organized as follows: in Section II will discuss some related works. Section III discusses existing manual flashing process and problem statement; Section IV gives you an idea what is Loader Cart System; in Section V, we describe our proposed solution or implementation methodology adopted in the project and outcomes of the proposed solution; Section VI discusses advantages and disadvantage of the proposed system; and conclusion of the article is in Section VII.

II. RELATED WORK

A. Auto Upgrade Platform Tool

Figure 1 illustrates the main interface of the Auto Upgrade Platform application. It is a radio’s firmware flashing tool. Normally this tool is used if repeatedly flashing operation is needed in order to test the stability of the memory flash IC of one particular radio. Before the tool can be operated, we need to specify the paths of firmware needed in an Excel spreadsheet as shown in Figure 2.

After each firmware flashing operation, Auto Upgrade Platform tool logs the flashing result, verification result together with the time stamping into a file.

This tool is good for stability tests but is not suitable for Loader Cart System configuration because there are 8 different radios need to be configured simultaneously instead of repeatedly flashing into ONE radio only. Moreover, Auto Upgrade Platform application will overwrite the original tunned data of each radio unless the tester reserves this data manually before firmware flashing is started.

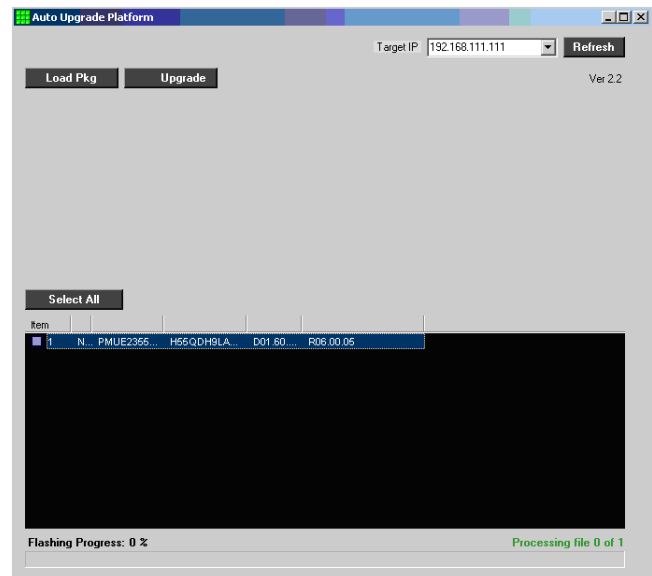


Figure 1. Main interface of Auto Upgrade Platform Tool

W	Version	CP	Version	RAM	DL	PPSDT	Path	Boot	Path	HAL	Path	Host	Path	CP1	Path	CP2	Path	CP3	Path
01.60.16	R06.00.05	\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0																	
01.60.16	R06.00.05	\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0																	
01.60.16	R06.00.05	\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0\Bqcg48-0																	

Figure 2. Firmware location specified in spreadsheet for Auto Upgrade Platform Tool

B. Flash Device Tool

This is a flashing tool that is almost similar to Auto Upgrade Platform Tool mentioned above. It requires the user to select or enter all the paths where firmware is located. This tool is not going to preserve the radio’s tunned data as well.

C. HP® QuickTest Professional™ [3]

HP® QuickTest Professional™ (QTP) supports functional and regression test automation that addresses every major software application environment. QTP uses keyword-driven testing concept to simplify test creation and maintenance. The most interesting feature of QTP is, it allows testers to build functional and regression test cases by capturing flows directly from the application screens using specialized capturing technology. Kindly refer to [4] for more information.

III. EXISTING PROCESS AND PROBLEM STATEMENT

A. Manual Process

The existing process is done manually as listed below. The flow listed is considered as 1 cycle:

1. Equipments setup.

2. Manually preserve and upgrade tuning data: This process requires extracting all records from the radio [5] and then clones or upgrades tuned data to the latest version of radio's firmware. This step requires ~17 minutes per radio in average.
3. Flashing: Flashing all required firmware into radios by using firmware flashing tool.
4. Radio's settings: Clone the radio's setting provided by the issue submitter into the radio and make necessary changes including serial number, IP address, etc. This step is done by using firmware upgrade/update and radio's settings editing tool.

In an average, the whole process required ~4 hours to configure 8 radios. This amount of time is just for one cycle and the time taken for debugging, implementation, and firmware compilation has yet to be considered. Normally, more than one cycle is needed for every issue as the developer may need to add in more debug code or upgrade the radio firmware when a new firmware is out, or re-test the firmware after the fix is applied to confirm the fix is work properly. Hence, reducing "non-value-added" appraisal works of developers is definitely needed.

IV. LOADER CART SYSTEM

Loader Cart System is an automation tool to stress the radios to its limit to test the radio's stability, robustness, and reliability. Some basic functionality such as making calls and sending messages can be found in this system. Now, testers just need to produce their own test cases that can potentially surface the defect of the radio's firmware. Figure 3 below illustrates the Loader Cart is connected to a computer. A computer software is developed in LabViewTM[6], a visual programming language, and used to communicate with the Loader Cart that connected to all 8 radios (device under test).

As we realize, automation tool is used in testing because testing can be carried out on 24*7 bases. This kind of testing is helpful to surface memory leak issues, system stability issues, etc.

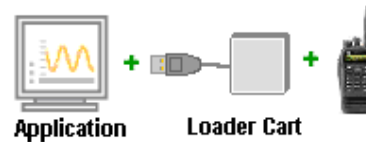


Figure 3. Loader Cart

V. PROPOSED SOLUTION AND OUTCOMES

The proposal is to automate the configuration process by using AutoIT. The aim is to minimize human's effort. Therefore, we could spend more time on value-added works. The solution is "integrating" all applications that needed such as package creation tool and firmware upgrade/update tool, Radio's Settings Editing Tool, and Microsoft Windows Command Prompt terminal as one automation tool. AutoIT is used to control the interaction of these applications without human attention. AutoIT acts as "human brain" in this automation process.

AutoIt [1] is a freeware designed for automating the Windows GUI and general scripting. It uses a combination of simulated keystrokes, mouse movement and window or control manipulation in order to automate tasks. AutoIt is also very small, self-contained and run on all versions of Windows with no annoying "runtimes" required. AutoIt was initially designed for PC "roll out" situations to reliably automate and configure thousands of PCs. Over time it has become a powerful language that supports complex expressions, user functions, loops and etc. AutoIT Scripts can be compiled into stand-alone executables as well. This is very useful as the script can be executed anytime and anywhere in this world. Kindly refer [1] for more details. Basically, AutoIT is a tool that is quite similar to QTP (QuickTest ProfessionalTM) from HP[®] (Hewlett-Packard[®]) in terms of behaviour. More attractively is no license is required for AutoIT. It is very feasible for a company to adopt when budget is the main concern.

Figure 4 illustrates the AutoIT IDE (Integrated Development Environment) or normally known as Script Editor in AutoIT which is quite similar to other text/code editor but with code compilation capability. All APIs (Application Programming Interface) are documented in AutoIT online

documentation [7] that looks like Javadocs [8], which can be found online easily. Any programmer can easily learn up the scripting language within a short period of time, such as within 1 or 2 days.

There are 5 main parts to be automated as listed below:

1. Editing IP address, radio's serial number, and radio's ID by using Radio's Settings Editing tool.
2. Package creation: All in one file creation.
3. Radios upgrading process by using firmware upgrade/ update tool
4. Preserve similar radio's settings as issue reported
5. Clear all old records in the radio before start testing

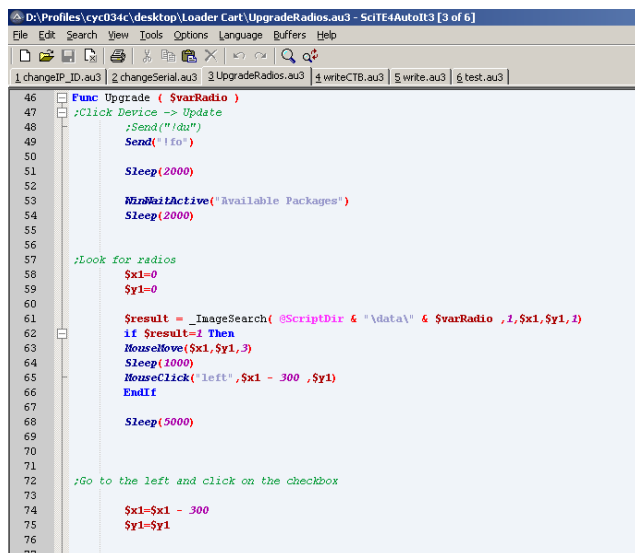


Figure 4. AutoIT Script Editor Windows (IDE)

The detail processes described as below:-

1) Edit Radio's Settings: This process includes changing IP address, radio's serial number, and radio's ID by using Radio's Settings Editing Tool. After that, all settings are stored in the local storage.

2) Upgrade Package Creation: This is to create a firmware package that includes all firmware that are needed for a radio to operate. This process allows the developers' debug firmware and packs it together with other necessary files such as radio's settings files from item 1 to become one file that easier to flash into the radio.

3) Update Radios or Radio Flashing: AutoIT automates firmware upgrade/update tool to flash the upgrade package created in item 2 into the radio. This method will preserve the existing tunned data of the radio.

4) Radio's Settings Cloning: This process is going to clone the similar radio's settings as issue reported into radio.

5) Old Records Clearance: Before every testing is started, clearing old reset record resides in radio is needed. This process is necessary so that the new reset record will not be corrupted by the old reset record. Normally this might cause misleading in issue root-causing.

After the configuration is done, the Loader Cart System is ready for tests-execution or issues reproduction. An engineer can start to reproduce the reported issue with debug firmware with Loader Cart after appropriate channel for each radio is set.

With AutoIT, we could eliminate process 2, 3, and 4 (in section III. Existing Process and Problem Statement) that was done manually in the past as AutoIT can control all applications that are needed in Loader Cart System configuration. Figure 5 gives you a big picture on how the automation tool works. This newly created automation tool will help to make engineer's life easier. Now, everything is just in ONE click. What developer should do is only to kick start the automation and leave it execute by itself. There is no human's attention needed during the configuration process (during automation execution). Engineer can proceed with Loader Cart System test-execution after the configuration process is done. Hence, this reduces engineer's "non-value-added" appraisal works significantly. According to our experiments, this tool is able to reduce engineer's operation time from ~4 hours to ~30 minutes in average based on every cycle of Loader Cart System configuration. 30 minutes basically is just for equipments setup such as equipment searching and miscellaneous setup including radio's channel setting.

Manual and automated flashing performances for 8 radios are shown in the Graphs of Figure 6.

Noticed, the automated flashing⁴ time is much lesser than manual flashing time which is ~2.7x less per cycle. This is because we had automated some configuration processes with AutoIT, such as editing radio's settings, upgrade package creation, radios flashing; cloning and editing radios' setting, and clearing all old records in the radios. In other word, with the enhancement, every cycle of Loader Cart Automation, we are able to further reduce our engineer's effort by ~85% as illustrated in Figure 7.

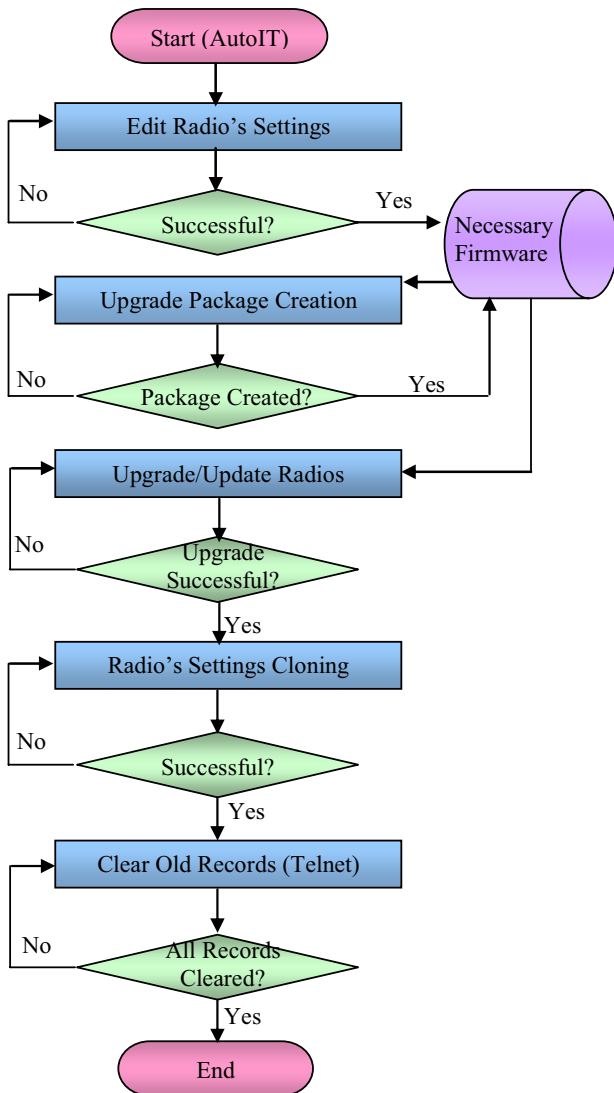


Figure 5. Automation Flashing Flow Chart

⁴ This is the flashing time done in automation, which does not need any attention from engineer.

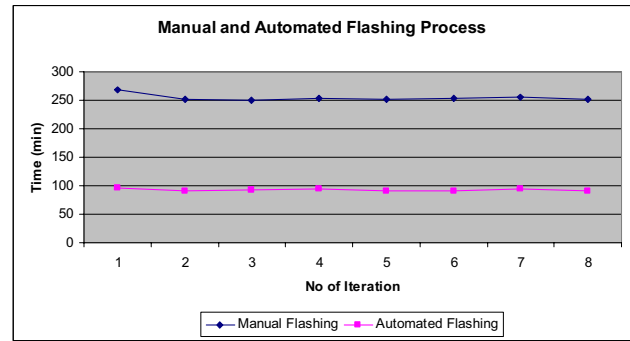


Figure 6. Manual and Automated Flashing Performance

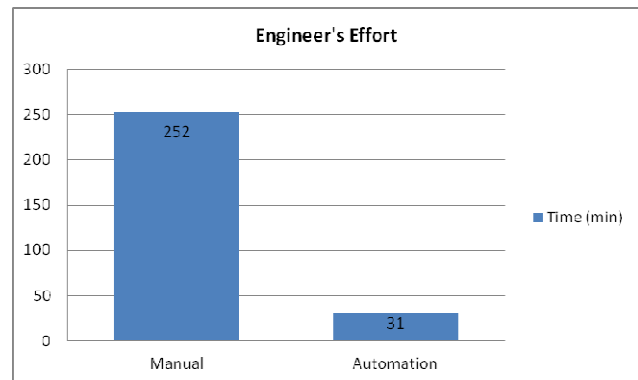


Figure 7. Engineer's Effort Saving (~85%) for Every Cycle of Loader Cart Automation

VI. ADVANTAGES AND DISADVANTAGES

A. Advantages

There are few advantages in this implementation

- No additional license purchasing: This is because AutoIT is a freeware compared to other license required tools. Hence, this proposal is suitable for the project with limited budget.
- Significantly reduces human's operation hours ~85% from ~4 hours to ~30 minutes. This benefits all developers that required setup a loader cart system in their issue reproduction.
- More time for value-added works such as debugging, code implementation, new feature development, looking/searching for new ideas, etc.
- Improving Loader Cart automation tests cycle time by ~63%.
- Less human's interaction: With this automation tool, user can start the configuration with ONE click besides equipments setup.

- AutoIT scripting is easy. Developer can easily obtain the necessary documentations from internet or online documentation from AutoIT homepage [6].
- AutoIT is compatible to most of operating systems that are available. The program can also be executed in any platform without any installation since AutoIT Scripts can be compiled into stand-alone executables.
- AutoIT has been designed to be as small as possible and stand-alone with no external dynamic link library files or registry entries required making it safe to use on Servers.
- This newly created automation tool is not limited for developers but also can be utilized by Integration Test (IT) team, and System Test team.
- When error met during AutoIT execution, an email can be triggered to notify developer so that immediate attention can be provided such as rectify the error to make sure testing progress is not been affected.
- More user friendly user interfaces are needed in order to improve overall user experience and further improve testing cycle time.
- Applying this project as a Green Belt project of Six Sigma and further data, details collection need to be conducted.

VIII. CONCLUSION

In short, with this newly proposed automation tool by using AutoIT, we are able to reduce ~85% “non-value-added” appraisal works of an engineer, from ~4hours to ~30minutes, which is a huge saving for a company. Engineer can fully focus on other research and development works with these “extra” hours. On the other hand, this tool requires no additional license purchasing, which potentially saves the company some money.

B. Disadvantage

The main disadvantage of this automation is as below:

- Once the automated configuration tool is started, no interruption is allowed such as using the same machine to launch some other application for other operation. This is because AutoIT is designed for automating the Windows GUI. The test will fail if AutoIT detects different GUI appeared on the machine’s screen. In this case, engineer needs to restart the configuration process again. Thus, a dedicated machine is needed for this purpose.

VII. FUTURE WORKS

There are few things can be done in the future as listed below.

REFERENCES

- [1] (1999-2000) AutoIT v3 Homepage,. [Online] Available: <http://www.autoitscript.com/autoit3/docs/introduction.htm>.
- [2] (2007) How Do I: Use the Hopper Test Tool To Stress Test My Windows Mobile Application? [Online]. Available: <http://msdn.microsoft.com/en-us/netframework/bb964490.aspx>
- [3] A. Jain, Best practices with QuickTest Professional™, [Online]. Available: <http://www.qtptraining.com/QTPOptimization.pdf>
- [4] HP® QuickTest Professional™ Software Data Sheet, HP®, 2008.
- [5] Teoh En Sheng, How To Backup Radio’s Flash, Motorola™ Internal Document Repository, Motorola™, 2007.
- [6] (2010) NI™ LabView™ website. [Online]. Available: <http://www.ni.com/labview/>
- [7] (2010) AutoIT Online Documentation : Function Reference. [Online]. Available : <http://www.autoitscript.com/autoit3/docs/functions.htm>
- [8] (2003) Java 2 Platform Enterprise Edition, v1.4, API Specification. [Online]. Available : <http://download.oracle.com/javase/1.4/api/>