How to build NOS/BE 1.5 Level 712 from Scratch

A Step-by-Step Guide to Configuring and Building NOS/BE 1.5 Level 712 from “Scratch”.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author:</td>
<td>Joachim Siebold</td>
</tr>
<tr>
<td>Date:</td>
<td>October, 2nd 2022</td>
</tr>
<tr>
<td>Cataloged:</td>
<td></td>
</tr>
<tr>
<td>Edited:</td>
<td>December, 15th 2022</td>
</tr>
</tbody>
</table>

1. Introduction

Configuring and building a NOS/BE deadstart tape from scratch is a tedious job. For a full system you have to run more than 40 batch jobs and mount dozens of tapes. Fortunately the recent developments of the DtCyber emulator provide a powerful programming interface to control both the emulator and the running operating system.

Therefore, I could realize a DtCyber environment, which supports building a NOS/BE deadstart tape with much software included with a high degree of automation.

The starting point is an unconfigured NOS/BE L650 deadstart tape, a L712 correction tape and a bunch of L650 product tapes which came from the Zentraleinrichtung Rechenzentrum Technische Universität Berlin (ZRZ TUB) which got a CDC CYBER 175 in 1976. These tapes include NOS/BE, FTN with TSFTN, advanced CRM, FTN5, PMD for FTN and FTN5, 8bit subroutines, FORM, SORT5, Fortran 4 to 5 Conversion Aid, Cyber Interactive Debug and Intercom 5 with RDF.

PASCAL 6000, the UMN Fortran 5 compiler (M77) and the UMN Interval Arithmetic Library are added from existing source tapes.

Other compilers, where sources do not exist according to current knowledge are extracted from a configured TUB deadstart tape. These include UMN Fortran IV (MNF), Algol 60, Algol 68 and PL/1.

The sources of additional products (DDL3, CDCS2, QU3, FDBF, DCL2, BASIC and COBOL5) are retrieved from NOS2 L871 source tapes and rolled back to a L650 compatible update status.

Unfortunately, the unconfigured L650 system is not capable to build a deadstart tape for all products because of system table limitations.

Therefore, building the full system uses three DtCyber environments:

1. Initialize, configure and run the NOS/BE L650 bootstrap system (DtCyber configurations: bootstrap-cti, bootstrap-level0, bootstrap) and create a deadstart tape for the NOS/BE L712 build system. The L712 build system has only the products installed which are needed to build the production system.
2. Initialize and run the NOS/BE L712 build system (DtCyber configurations: build-cti, build-level0, build) and create a deadstart tape for the production system. You can choose, if you would like to build either a system with or without extended memory support.
3. Initialize and run the NOS/BE L712 production system without EM (DtCyber configurations: nosbe-cti, nosbe-level0, nosbe).
4. Initialize and run the NOS/BE L712 production system with EM (DtCyber configurations: nosbe-em-cti, nosbe-em-level0, nosbe-em).

There is a zip archive for a ready-to-run NOS/BE L712 build system. Therefore, you can shorten the build process considerably.

In this document, the following markups are used:

<table>
<thead>
<tr>
<th>Markup</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(CR)</td>
<td>Carriage Return</td>
</tr>
<tr>
<td>(BS)</td>
<td>Backspace</td>
</tr>
<tr>
<td>=XYZ=</td>
<td>Input of “XYZ” at the DtCyber Console</td>
</tr>
<tr>
<td><em>XYZ</em></td>
<td>Input of “XYZ” to DIS at the DtCyber Console</td>
</tr>
<tr>
<td>(operator) XYZ</td>
<td>Input of “XYZ” at the DtCyber operator prompt</td>
</tr>
<tr>
<td>(shell) XYZ</td>
<td>Input of “XYZ” at a shell prompt</td>
</tr>
<tr>
<td># XYZ</td>
<td>XYZ is a comment</td>
</tr>
</tbody>
</table>

There are a number of JavaScript files (suffix: js) in the NOSBE712 directory. These scripts are called in Linux/mac OS and Windows in different ways.

Linux, mac OS:

./scriptname.js <parameters>

Windows:

node scriptname <parameters>

In this document, the Linux/mac OS calling style is consistently used.

2. What you need

1. The DtCyber emulator. At least version 5.7.24 is required.
2. A current software version of Node.js.
4. The NOS/BE 1.5 Level 650 unconfigured software kit (nosbe650.zip).
5. The file CIP860V700.TAP, which is in the cipassortment.zip archive.
6. The othertapes.zip archive.
7. A C-Compiler, if you would like to use the NOS/BE print postprocessor (see below).
8. A recent Linux, mac OS or Windows system.
3. Prepare the build environment

1. Unzip the file NOSBE712-main.zip either in the root directory of the DtCyber software or at another location. In the latter case you must define the environment variable DTCYBER pointing to the root directory of the DtCyber software. Rename the directory name NOSBE712-main to NOSBE712.
2. Create a directory NOSBE650 at the same level as the NOSBE712 directory and store all tape image files from the archive nosbe650.zip there.
3. Create a directory CIPTAPES at the same level as the NOSBE712 directory and store the file CIP860V700.TAP from the archive cipassortment.zip there.
4. Create a directory OTHERTAPES at the same level as the NOSBE712 directory and unzip othertapes.zip there.
5. If you would like to use the NOS/BE print postprocessor, go to the subdirectory NOSBE712/lpt2pdf and build the lpt2pdf executable:

   (shell) gcc -o lpt2pdf lpt2pdf.c (Linux)
   or
   (shell) cl lpt2pdf (Windows)

4. Description of the build environment

The DtCyber configuration file cyber.ini provides separate environments (disk and CM/ECS storage) for the NOS/BE 1.5 L650 bootstrap system, the NOS/BE L712 build system and the NOS/BE 1.5 L712 production system. They use the following deadstart tapes:

1. NOS/BE L650 bootstrap system: OTHERTAPES/DSTAPE.tap
2. NOS/BE L712 build system: DeadstartTapes/DSTAPE_BUILDSYS.tap
3. NOS/BE L712 production system without EM: DeadstartTapes/DSTAPE.tap
4. NOS/BE L712 production system with EM: DeadstartTapes/DSTAPE_EM.tap

All deadstart tapes have no cti prepended. Thus the deadstart process is as follows:

1. Deadstart from the CIP tape and install CTI on disk. You have to do this only once.
2. Deadstart CTI from disk, then instruct CTI to deadstart NOS/BE from tape

The main component of the build environment is the autojob.js program. This program runs a series of batch jobs one after another. For each batch job it issues the necessary DtCyber operator and DSD commands to ensure an automatic tape assignment, then it submits the card deck and waits for the batch job to complete. If the job failed, autojob.js terminates (configurable!), otherwise the next batch job is executed.

The series of batch jobs is specified in configuration files, which are located in the node_modules subdirectory:

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>build_nosbe_buildsys</td>
<td>creates deadstart tape scratch/DSTAPE_BUILDSYS.tap for the L712 build system</td>
</tr>
<tr>
<td>build_nosbe</td>
<td>creates deadstart tape scratch/DSTAPE.tap for the L712 production system (without EM)</td>
</tr>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>build_nosbe_em</td>
<td>creates deadstart tape scratch/DSTAPE_EM.tap for the L712 production system (with EM)</td>
</tr>
<tr>
<td>run_postinst</td>
<td>post install jobs for the L712 production system (without EM)</td>
</tr>
<tr>
<td>run_postinst_em</td>
<td>post install jobs for the L712 production system (with EM)</td>
</tr>
</tbody>
</table>

You find a detailed description of an autojob.js configuration file in the file node_modules/run_tests.js. Job decks for building NOS/BE are located in the buildjobs directory.

Note: the NOS/BE system build jobs are designed for a single processor configuration, because file system corruptions happened with a dual-processor NOS/BE system.

If you would like to build a dual processor version, you can enable the IP.NCPU parameter in the 07_pla1.txt and 07_pla1_em.txt job decks. You were warned.

There are some helper scripts in the NOSBE712 directory that ease the control of the DtCyber emulator, e.g. for tape handling, batch processing and system shutdown. See the How to use a ready to run NOS/BE L 712 system.

5. Description of the NOS/BE print postprocessor

NOS/BE creates printer files with ANSI printer control. Form feed and line spacing are controlled by characters in column 1 of the printer file. The print post processor reads continuously the output of the printer file LP5xx_C12_E5, interprets form feed an line spacing controls, separates the output into single print jobs and calls the program lpt2pdf (Author: Tim Litt) to create a nice high-fidelity fan fold paper PDF file for each job. The PDF files are stored in the printfiles directory with time-stamped file names.

Note: If you would not like to use the print post processor, you should disable the helpers entries in the cyber.ini file.

6. Bootstrapping the Build System

If you would like to start with a ready-to-run build system, you can proceed to chapter 7.

6.1 Prepare the bootstrap system

If you would like to use a ready-to-run bootstrap system, you can unzip the archive nosbe650bootstrap.zip in the NOSBE712 folder and proceed to 6.2.

First you have to install CTI on disk:

```
(shell) ./dtcyber bootstrap-cti
=B= BUILD DEADSTART DISK
=M= MANUAL OPERATION
```
You have now to do a level 0 deadstart of the unconfigured NOS/BE Level 650 system. The deadstart tape does not contain any EST configuration, so you have to enter it manually. You also have to initialize the disk drives.

```
(shell) ./dtcyber bootstrap-level0
=O= OS LOAD WITH INTERVENTION
=S= SELECT DEADSTART DEVICE
=T= TAPE DEADSTART
=(CR)= ENTER TAPE TYPE
=(CR)= CHANNEL - 13
=(CR)= EQUIPMENT - 0
=(CR)= UNIT - 00
=4.Y(CR)= EQUIPMENT CHANGES
=5.N(CR)= EM OFF
=(CR)=
=1,AJ,C00,E0,U0,SY,S,P,F,Q,MD,SN=SYSTEM,V=SYS001(CR)=
=2,AJ,C00,E0,U1,PF,SN=SYSTEM,V=SYS002(CR)=
=20-23,NT,C13,U00+,A,G(CR)=
=27,DS,C10,E7(CR)=
=30,DC,C07(CR)=
=31,LP,C12,E5(CR)=
=32,CR,C12,E3(CR)=
=33,CP,C12,E6(CR)=
=SYSTEM,SYS,PF,Q,SCR(CR)=
=SYSTEM,I(CR)= # initialize disks
=(CR)=
=(CR)= RMS CONTROLLERS
=GO(CR)= DST 02 CONTROLWARE CONFLICT -LOAD CONTROLWARE
=FT(CR)= LABEL INITIALITATION
=ALL(CR)= # Wait for system load to complete *
=0010185(CR)= # enter date
=000000(CR)= # enter time
=AB.(CR)=
```

6.2 Create the NOS/BE L712 Build System

Do a level 3 deadstart of the bootstrap system:

```
(shell) ./dtcyber bootstrap
```
Preconfigured key entries in the operator section provide an automatic deadstart.

Now you start the configuration and generation for the NOS/BE L712 build system:

```
(shell) ./autojob.js build_nosbe_buildsys
```

This takes about 2 hours on a Raspberry Pi4 (Debian 64bit) to complete. The new deadstart tape is stored as `scratch/DSTAPE_BUILDSYS.tap`. You must now copy `scratch/DSTAPE_BUILDSYS.tap` to `DeadstartTapes/DSTAPE_BUILDSYS.tap`

### 7. Build the production system

If you skipped chapter 6 unzip the archive `nosbe712build.zip` in the `NOSBE712` folder and proceed to 7.2 or 7.3.

#### 7.1 Prepare the Build System

First you have to install CTI on disk:

```
(shell) ./dtcyber build-cti  
(see 6.1)
```

Now do a level 0 deadstart of the NOS/BE Level 712 build system. You also have to initialize the disk drives. An EST is already compiled into the deadstart tape.

```
(shell) ./dtcyber build-level0  
=0=  OS LOAD WITH INTERVENTION  
=S=  SELECT DEADSTART DEVICE  
=T=  TAPE DEADSTART  
=(CR)= ENTER TAPE TYPE  
=(CR)= CHANNEL - 13  
=(CR)= EQUIPMENT - 0  
=(CR)= UNIT - 00  
=GO(CR)= TO USE 16B PPUS  
=4.Y(CR)= EQUIPMENT CHANGES  
=(CR)=  
=SYSTEM,I(CR)= # initialize disks  
=(CR)=  
=(CR)= RMS CONTROLLERS  
=FT(CR)= LABEL INITIALIZATION  
=ALL(CR)= # Wait for system load to complete *  
=00100191(CR)= # enter date *  
=T000000(CR)= # enter time *  
=AB.= # system / cp display  
(shell) ./shutdown.js
```
7.2 Build the NOS/BE L712 Production System (without EM)

Do a level 3 deadstart of the build system:

```
(sh) ./dtcyber build
```

Preconfigured key entries in the operator section provide an automatic deadstart.

Now you start the NOS/BE system configuration and generation for the production system:

```
(sh) ./autojob.js build_nosbe
```

This takes about 7 hours on a Raspberry Pi4 (Debian 64bit) to complete. The new deadstart tape is stored as `scratch/DSTAPE.tap`. Copy that file to `DeadstartTapes/DSTAPE.tap`.

7.3 Build the NOS/BE L712 Production System (with EM)

Do a level 3 deadstart of the build system as described in 7.2. Then start the NOS/BE system generation for the EM enabled production system:

```
(sh) ./autojob.js build_nosbe_em
```

This takes about 7 hours on a Raspberry Pi4 (Debian 64bit) to complete. The new deadstart tape is stored as `scratch/DSTAPE_EM.tap`. Copy that file to `DeadstartTapes/DSTAPE_EM.tap`.

8. Initialize the NOS/BE L712 Production System

8.1 Production system without EM

Install CTI on disk for the production system:

```
(sh) ./dtcyber nosbe-cti
```

(see 6.1)

Then do a level 0 deadstart:

```
(sh) ./dtcyber nosbe-level0
```

(see 7.1)

Start the queue manager and see if JANUS is running on CP1.

```
=AUTO.=
```

Now run the post installation jobs. These jobs set up the Intercom configuration and restore some
system files from the build system. You have to confirm the Intercom configuration changes at the DtCyber console.

(shell) ./autojob.js run_postinst
=20.GO= ATTEMPT TO ALTER INTERCOM CONFIGURATION
=20.GO= ATTEMPT TO ALTER INTERCOM PASSWORDS
* wait for all batch jobs to complete
(shell) ./shutdown.js

### 8.2 Production system with EM

Install CTI on disk for the production system:

(shell) ./dtcyber nosbe-em-cti
(see 6.1)

Then do a level 0 deadstart:

(shell) ./dtcyber nosbe-em-level0
(see 7.1, but in addition you have to initialize the EM if you run this system the first time or altered the EM config)

...=
=4.Y(CR)= EQUIPMENT CHANGES
=5.I(CR)=
=(CR)=
=60(CR)= EM PARTITIONING
=(CR)= EM PARAMETERS
...

Start the queue manager and see if JANUS is running on CP1.

=AUTO.=

Now run the post installation jobs. These jobs set up the Intercom configuration and restore some system files from the build system. You have to confirm the Intercom configuration changes at the DtCyber console.

(shell) ./autojob.js run_postinst_em
=20.GO= ATTEMPT TO ALTER INTERCOM CONFIGURATION
=20.GO= ATTEMPT TO ALTER INTERCOM PASSWORDS
* wait for all batch jobs to complete
(shell) ./shutdown.js

### 9. Running the NOS/BE L712 production systems
See How to use a ready to run NOS/BE L712 system for details.

10. Contributors

William Schaub solved a serious bug in the CDC SORT/MERGE build scripts, which was difficult to fix. Kevin Jordan improved the DtCyber configuration.

11. Terms of Use

This wiki page provides CDC copyrighted material, which may only be used in accordance with the BT Federal Permit.