Using an enhanced CDCF format for custom CANopen device testing and configuration

Bernhard Floeth, Adam Opel AG
Olaf Pfeiffer, Embedded Systems Academy GmbH

The "Concise Device Configuration File" format was conceived to support a simple method to configure a CANopen device. The CDCF is primarily a list of write accesses for an Object Dictionary.
This paper describes an enhanced CDCF format which supports various commands including setting timeouts, reading back values for confirmation, transmitting the NMT master message and executing a LSS Master cycle. These commands support custom test sequences including identification. A player supporting this format can verify if a device matches the CDCF (for example vendor ID and product code match), before continuing. The files required can be created based on CSV files as supported by spread sheet programs, which greatly simplifies CDCF generation and editing.

CDCF Introduction

The "Concise Device Configuration File" is specified in CiA 302-3 [1]. In short, it is an array of records with the following contents:

Index (type UNSIGNED16)
Subindex (type UNSIGNED8)
Length (type UNSIGNED32)
Data (type DOMAIN – any data, length)

A CANopen Master or test utility can process this list and "execute" it as a sequence of SDO (Service Data Object) write accesses to a specific node, the device under test (DUT). In traditional CANopen systems, this file format can be used to configure a CANopen node, for example setting specific heartbeat and/or PDO (Process Data Object) transmission modes or event times.

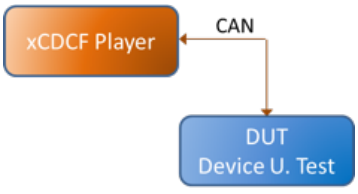

Figure 1: Stand-alone CDCF usage

The CDCF file can typically be applied by a CDCF player in both stand-alone mode or during regular (pre)operation. In stand-alone mode, only the player and the device to be configured are connected to CAN, as illustrated in figure 1.

The use case illustrated in figure 2 shows how a CDCF player can also operate during regular operation. However, in this case the operator needs to ensure that the CANopen SDO channel required to communicate with the Device Under Test (DUT) is not used by another device in the system.

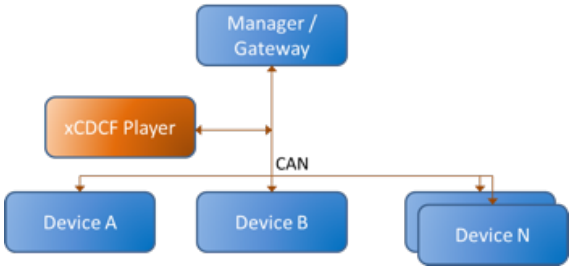

Figure 2: CDCF usage during operation

Some application profiles such as CiA 447 support dedicated SDO channels and leave the default CiA 301 SDO channels unused. In such a scenario, a CDCF player can operate using the regular CiA 301 SDO channels at any time without the risk of collisions due to multiple use of the same SDO channel / CAN IDs.

CDCF "Execution"

Upon start, a CDCF Player needs to be told (for example by configuration or interactively by a user) which file should be executed towards which SDO channel (SDO client commands send to which node ID).

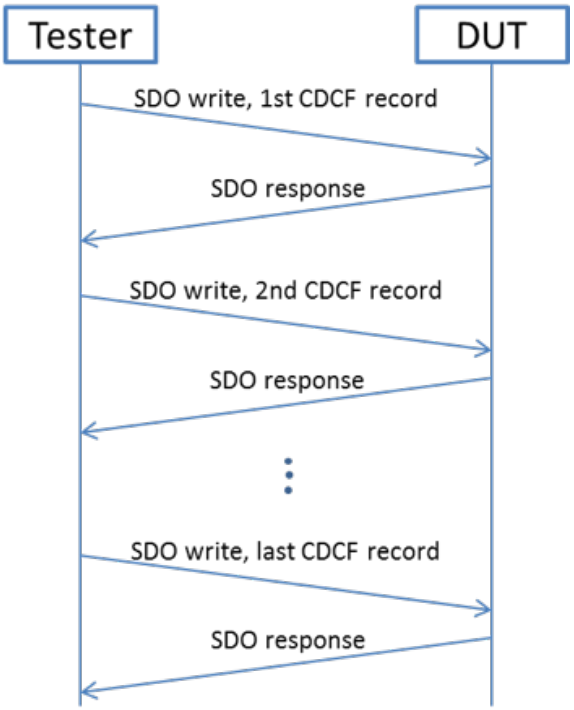

Figure 3: CDCF write process

The player sends the entries in the CDCF one-by-one as a SDO client command to the selected node ID as illustrated by figure 3. Each transmitted SDO client write access requires a SDO response from the DUT for the next step to be executed.

Drawbacks of CDCF usage

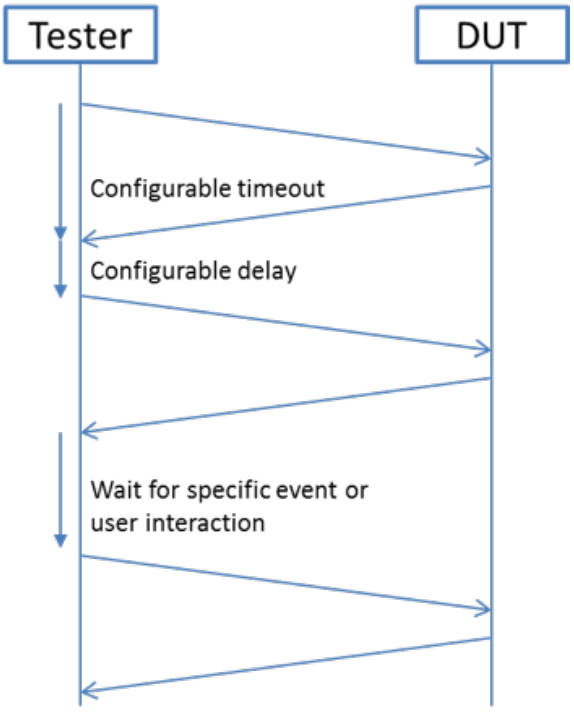

Figure 4: Possible interactions

As is, the CDCF is just a list of data for Object Dictionary entries. There is no timing information (how fast to execute the sequence) or device information (to identify if this file is for a specific device only) or flow control (only continue if the CANopen device has specific values at selected Object Dictionary entries).
There are no physical layer settings associated with a CDCF: CAN bitrate used, timeouts and delays for SDO transfers or the node ID of the device this file is intended for.

Requirements for custom test sequences

In order to make the CDCF usable for customizable test sequences or device identification, new commands need to be specified.

File identification, version, comments
These allow an identification of a specific CDCF. Name, comments and version information can be added.

CDCF Player Settings
Support setting of CAN bitrate, node ID, SDO timeouts and delays. Configure number of retries on failures and logging options for debugging and test of the CDCF.

Active Controls
Define pauses or wait for action of the selected device, such as a bootup message or an operational heartbeat. Initiate the execution of a LSS Master Cycle or an NMT Master message. Execute a SDO read access instead of a write access.

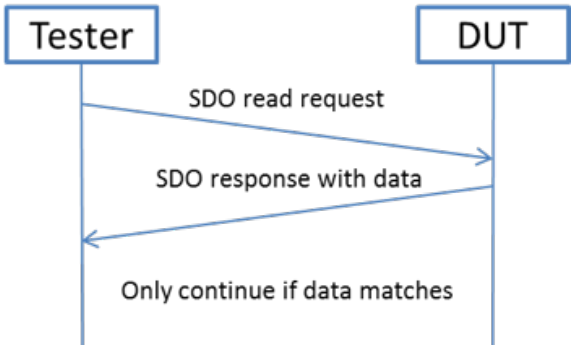

Figure 5: Verification of entries

Support of reading and verifying the contents of Object Dictionary entries is an essential new feature to ensure that the selec-

ted target DUT meets the requirements for the following configuration.

Command definition

To keep the new CDCF enhancements backward compatible, all new commands should be “hidden” in the existing regular records with Object Dictionary entries. The extend commands defined in this document stick to the existing record format. An Index value of 0F0Fh is used to identify enhanced commands. In CANopen this index value is currently reserved [2].
Whenever the device executing the records of a CDCF reaches a record with the Index 0F0Fh, it does not immediately generate an SDO write but interprets the contents as an extended command to execute and processes it accordingly.
A CDCF executing device that is not capable of interpreting these commands will generate an SDO write to Index 0F0Fh which will result in an SDO Abort as 0F0Fh is a reserved value in the CANopen Object Dictionary.

List of new commands

Informational Strings
These strings do not have any effect on the CANopen communication. They are used to identify the file or offer additional progress or debug information. If a CDCF player has a display, these can be shown to the user while the CDCF file is processed.

[0F0Fh,01h] VISIBLE_STRING,
File Information:
String with information about this file, no further effect.

[0F0Fh,02h], VISIBLE_STRING,
Conditional Error Info:
String with error information, no effect, just treated as a comment. This error is considered “occurred” if the PREVIOUS record in the CDCF produced an error (could not handle or abort returned).
[0F0Fh,03h], VISIBLE_STRING,
Comment:
String with a comment.

[0F0Fh,01h], VISIBLE_STRING,
User action:
String to display to user and wait for user action. This could be a message to the user like “now power cycle the device”.

CDCF Player Settings
This group of commands specifies settings that are typically made within a CDCF player, like selecting a specific node ID to which the SDO requests are send or setting timeouts.

[0F0Fh,11h], UNSIGNED8,
Set Bit Rate:
If the bit rate is known, it may be specified here. Values are as defined by [3].
FFh: use default of player
0: 1 Mbit/s
1: 800 kbit/s
2: 500 kbit/s
3: 250 kbit/s
4: 125 kbit/s
5: reserved
6: 50 kbit/s
7: 20 kbit/s
8: 10 kbit/s
9-FEh: reserved

[0F0Fh,12h], UNSIGNED8,
Set Node ID:
The CDCF file containing this command is intended for the node ID (1-127) specified.
FFh: use default of player
1-7Fh: use this node ID
80h-FEh: reserved

[0F0Fh,13h], INTEGER8
Next Node ID Offset:
Add this value to the currently used default node ID. This allows working on different nodes or nodes which change their node ID, for example when in bootloader mode.

[0F0Fh,14h], UNSIGNED16,
SDO timeout:
Sets the SDO timeout used for the read/write accesses to the value passed (in ms).
FFFFh: use default setting of player

[0F0Fh,15h], UNSIGNED16,
Back to back delay:
Sets the delay used between processing individual records in the CDCF (in ms).
FFFFh: use default of player

[0F0Fh,16h], UNSIGNED8,
Maximum retries:
If an SDO access fails (Abort or wrong data read), then retry up to this maximum.
FFh: use default of player

[0F0Fh,17h], UNSIGNED8,
Logging detail:
Enables/disables the generation of a log file by the CDCF player, if supported.
0: disable
1: enable – detail level minimum
2: enable – detail level plain
3: enable – detail level detail
4: enable – detail level debug
5-255: reserved

Active Control
These commands allow activating controls in the CDCF player including NMT Master functionality.

[0F0Fh,21h], UNSIGNED16,
Pause/delay:
Pause execution of the CDCF player for this delay (in ms).

[0F0Fh,22h], UNSIGNED8,
Wait for action:
0: wait for a bootup of node
5: wait for node to be operational
7Fh: wait for node to be pre operational
FFh: wait for any next heartbeat of node
All other values are reserved.
[0F0Fh,23h], UNSIGNED16,
Request execution of NMT command:
Request generation of a NMT command, the 16bit value contains the 8 bit NMT command (bit 0-7) 8 bit Node ID (bit 8-15).

[0F0Fh,24h], LSSMASTERRECORD,
Run a LSS Master Cycle:
Starts a LSS Master cycle, the LSSMASTERRECORD contains
UNSIGNED32 Vendor ID
UNSIGNED32 Product Code
UNSIGNED32 Revision
UNSIGNED32 Serial Number
UNSIGNED8 Node ID to assign
A value of all bits set means “use default” of CDCF player or do not care if not further specified.

[0F0Fh,25h], UNSIGNED8,
Execute SDO Read:
The next CDCF record is NOT executed as an SDO write. It is executed as an SDO read. Data is read into a local buffer (can later be used by subindex 26h)
Bit 0: match / no match - ignore data returned or verify it matches
Bit 1: wait for – if data returned does not match then wait for repetition timeout and try again, maximum of retries (Subindex 4)
Bit 2-7: reserved

[0F0Fh,26h], UNSIGNED8,
Execute SDO Write from buffer:
The next CDCF record gets executed, but with data inserted from the local buffer, this allows writing back data previously read. Data field unused, set to FFh.

Usage example: Identification of device

A CDCF can now be associated with a specific device or device class by matching up its Vendor ID, Product Code, Revision information or even serial number. Execution aborts, if the desired entries do not match. For both test or configuration sequences this ensures that a CDCF only gets executed on those devices it was generated for. A CDCF can be generated to only match and work with an individual device (match down to serial number) or any device from a series (match to vendor ID, product code and possibly revision number).

Usage example: Test sequences

Supporting user interactions allows end of production line testing for output devices such as roof bars for police cars as defined by CiA 447 [4]. After each switch of an output, a user interaction like “verify if light xyz is now on” can be displayed on the CDCF player. The person running the test can now manually trigger execution of the next sequence by pushing a button/dial on the CDCF player.

Usage example: Safe and custom configurations

Some devices require a dedicated sequence in order to enable a custom

configuration mode. This might involve reading an entry and writing it back. With the enhanced functions a CDCF player can verify that the device under configuration is truly the device expected (read and verify selected entries), as well as execute custom sequences to activate a possible custom configuration mode.

Usage example: Boot loading

A CDCF can now contain all data and commands for boot loading a specific device. First a match verifies that this is the correct device, then the bootloader gets activated and last the sequence is executed to re-program/flash the firmware in the target device. This allows sending an “end user” device specific firmware update files that cannot accidently be programmed into the wrong device as it can be associated to the specific device the user has in his system. If needed, verification can go down to the serial number, so that a CDCF containing new code only works for one device with a specific serial number.

File generation and editing

Embedded Systems Academy provides a converter utility program that allows generating a CDCF from a comma separated value file as creatable with any spread sheet program such as Microsoft Excel. It simply contains the columns Index, Subindex and Data, the length information gets automatically calculated and inserted. If the data is too big to fit into the data column, the entry can also refer to a file that later gets inserted as data for this entry.

Availability

A free CDCF player supporting the commands listed in this paper is provided by Embedded Systems Academy. The free “CANopen File Player” can be downloaded from their web page and can directly execute .csv files.
In addition, enhanced CDCF players are implemented within the CANopen Diag systems.
All implementations support setting default parameters (for example for bit rate, timeouts and delays) as well as selecting one of multiple CDCF files stored on the system. User messages and interactions are supported and for each execution a log file can be created documenting the progress of execution of the CDCF.
The system is used by suppliers to Opel/GM to test their CiA 447 devices. In CiA 447 CDCF based test can be executed on individual devices under test (no other devices connected to the tester) as well as on devices in a CiA 447 system (testing on a live network).

References

[1] CiA 302-3, Additional application layer functions, Configuration and program download
[2] CiA 301, CANopen application layer and communication profile
[3] CiA 305, Layer Setting Services (LSS) and protocols
[4] CiA 447, Car add-on devices

Bernhard Floeth
Adam Opel AG
Bahnhofsplatz 1
65423 Rüsselsheim

Olaf Pfeiffer
Embedded Systems Academy GmbH
Bahnhofstraße 17
30890 Barsinghausen
Tel.: +49-5105-582-7897
Fax: +49-5105-584-0735
opfeiffer@esacademy.de
www.esacademy.com