POSITIONING AND TRACK CONTROL SYSTEM
ADDIPOS APCI-8001

NCC DLL

Rev. 3/082006

1 Introduction

The NCC compiler for compiling SAP programs for the PA8000, PS-840 and APCI-8001 and control systems is available as a DLL for Windows 32 systems. This enables users to compile source text files in rw_SymPas or G code file form to DIN 66025 into the intermediate code CNC binary files needed for the above systems, in their own programs.
To be able to execute these files, they must be transferred to the control unit concerned [txbf() / txbf2()] using the established methods, and started [startcnct()].

2 Using NCC.DLL

The NCC.DLL library provides various functions to parameterize and execute a compiling process. The system file SYSTEM.DAT is required for the compiling process. "SYSTEM.DAT" from the working directory is used by default. The source text file to be compiled is also required. The CNC output file is always stored in the same directory.
The compiling process must always be checked to see if it was successful. If an error is returned, the following error information can be read: File name, error line and error text. The file name is necessary since the error may also have occurred in an Include file.

3 Deliverables

The following software elements are supplied to use the NCC.DLL functions:

NCC.DLL	
NCC.H	
NCC.LIB	for various C compilers
NCC DLL.PDF	this document

4 Functions in NCC.DLL

4.1 CncCompile – compile G code file

DESCRIPTION:	This function compiles a G code file. The resultant binary file is given the extension .CNC, and is stored in the same directory as the source text file.
C:	int CncCompile (char *SrcFileName, int NumberAxis);
PARAMETERS:	SrcFileName is the name (including drive and path) of the source text file. NumberAxis contains the number of axes to be compiled for the program.
RETURN VALUE:	0 in the event of success, error number if an error occurs during the compiling process.
NOTE:	The number of axes in fact present is returned in the TOSI data structure when initialized with InitMcuSystem3() with a booted system. In certain cases (if a file is supposed to be the same for different axis configurations) it may be necessary for a file to be compiled for more than the existing axes. In this case a higher number of axes can be transferred. This corresponds to the "Full system" option with the other NCC versions. But here additional syntax errors can arise relating to axis naming. The axis names are taken from the file SYSTEM.DAT, and must be entered correctly for all axes used (mcfg.exe). G code files are always compiled for TASK 3.
EXAMPLE:	CompileError = CncCompile ("Example.SRC", 3);

4.2 CncSyntaxCheck – Syntax check at G-Code file

DESCRIPTION:	Command is the same as CncCompile (), however no binary file is generated.
C:	int CncSyntaxCheck (char *SrcFileName, int NumberAxis);

4.3 GetErrFileName – get file name in case of error

DESCRIPTION:	This function can be called up in the event of an error, to specify the compiling error.
C:	void DLLFUNC GetErrFileName (char **File name*);
PARAMETERS:	Pointer to a char array
RETURN VALUE:	None
NOTE:	The function writes the file names including drive and path to the first error arising in *File name*. *File name* must point to an area of memory that is big enough to take the file name (max. 260 characters).
EXAMPLE:	GetErrFileName (*File name*);

4.4 GetErrLine – get error line

DESCRIPTION:	This function can be called up in the event of an error, to specify the compiling error.
C:	int GetErrLine (void);
PARAMETERS:	None
RETURN VALUE:	Line number in which the first compiling error was detected.
NOTE:	The line number relates to the file whose name was returned with GetErrFileName().
EXAMPLE:	Error line = GetErrLine ();

4.5 GetErrText – get error text

DESCRIPTION:	This function can be called up in the event of an error, to specify the compiling error.
C:	void GetErrText (char * *Error text*);
PARAMETERS:	Pointer to a char array
RETURN VALUE:	None
NOTE:	The function writes a comment on the first error occurring in *Error text*. *Error text* must point to an area of memory that is big enough to take the error message (max. 256 characters).
EXAMPLE:	GetErrText (*Error text*);

4.6 GetSymAxisName – Determining the symbolic axis name

DESCRIPTION:	With this function the symbolc axis name of an axis can be read.
C:	int GetSymAxisName (int an, char *SymAxisName);
PARAMETER:	*an* is the index of the axis to be read. In *SymAxisName* the symbolic name, which has been read from the system file SYSTEM.DAT, is returned as string terminating on 0.
RETURN VALUE:	The indicated axis-index at success, -1 at failure
NOTE:	The storage space in *SymAxisName* must be large enough to store the axes name including the concluding zero sign. The maximum size is 16 bytes.
EXAMPLE:	char AxisName[16]; rvalue = GetSymAxisName (0, AxisName);

4.7 SapCompile – compile rw_SymPas file

DESCRIPTION:	This function compiles a rw_SymPas file. The resultant binary file is given the

	extension .CNC, and is stored in the same directory as the source text file.
C:	int CncCompile (char *SrcFileName, int NumberAxis);
PARAMETERS:	SrcFileName is the name (including drive and path) of the source text file. NumberAxis contains the number of axes to be compiled for the program.
RETURN VALUE:	0 in the event of success, error number if an error occurs during the compiling process.
NOTE:	The number of axes in fact present is returned in the TOSI data structure when initialized with InitMcuSystem3() with a booted system. In certain cases (if a file is supposed to be the same for different axis configurations) it may be necessary for a file to be compiled for more than the existing axes. In this case a higher number of axes can be transferred. This corresponds to the "Full system" option with the other NCC versions. But here additional syntax errors can arise relating to axis naming. The axis names are taken from the file SYSTEM.DAT, and must be entered correctly for all axes used (mcfg.exe).
EXAMPLE:	CompileError = SapCompile ("Example.SRC", 3);

4.8 SapSyntaxCheck – Syntax check at rw_SymPas file

DESCRIPTION:	Same command as SapCompile(), however without generating a binary file.
C:	int SapSyntaxCheck (char *SrcFileName, int NumberAxis);

4.9 SetCaseInsensitive - Differentiation capitalization/lower case printing on/off

DESCRIPTION:	For the translation of G-Code files the differentiation between capitalization/lower case printing can be disabled with this function.
C:	void DLLFUNC SetCaseInsensitive (int CaseInsensitive);
PARAMETER:	0 or 1 (default is 0)
RETURN VALUE:	None

4.10 SetSystemDatName – set system file

DESCRIPTION:	This function can specify the system file (drive, path, file name) with which the compiling process is carried out.
C:	void SetSystemDatName (char *str);
PARAMETERS:	File name with optional drive and path information
RETURN VALUE:	None
NOTE:	If this function is not called, the file "SYSTEM.DAT" is selected in the working directory.
EXAMPLE:	SetSystemDatName (“C:\Temp\System.dat”);

4.11 SetTaskNum – set task number

DESCRIPTION:	This function can specify the task number for which the source text file is compiled. This is only advisable when compiling rw_SymPas files, since G code files are always compiled for task 3.
C:	
PARAMETERS:	
RETURN VALUE:	
NOTE:	If the task is selected in the source text file with {$TASK ?} this instruction is void. The default value is 0.
EXAMPLE:	

4.12 SetTrac – set default track acceleration

DESCRIPTION:	This function sets the default track acceleration.
C:	void SetTRAC (double track);
PARAMETERS:	Acceleration value in the interpolation units currently set
RETURN VALUE:	None
NOTE:	See also SetTrvl and SetUnits

4.13 SetTrvl – set default track speed

DESCRIPTION:	This function sets the default track speed.
C:	void SetTRVL (double trvl);
PARAMETERS:	Speed value in the interpolation units currently set
RETURN VALUE:	None
NOTE:	See also SetTrac and SetUnits

4.14 SetUnits – set default travel and time unit

DESCRIPTION:	This function defines the default interpolation units. Without this command line, the position unit is set to mm and the time unit to seconds.
C:	void SetUnits (int pu, int tu);
PARAMETERS:	Position unit pu and time unit tu
RETURN VALUE:	None
NOTE:	See also PHB handbook, chapter "ctru, change trajectory units“
EXAMPLE:	SetUnits (0, 1); // select units mm and min

4.15 UseLineNumbers – line numbering on / off

DESCRIPTION:	This function can deactivate the need for line numbers (Nxxx) for compiling G code files.

C:	void DLLFUNC UseLineNumbers (int UseLineNum);
PARAMETERS:	0 or 1
RETURN VALUE:	None