**TC2中自动更换PLC运行程序并重启**

|  |  |
| --- | --- |
|  | 作者：夏璐杰  职务：华东区 技术工程师  邮箱：lj.xia@beckhoff.com.cn  日期：2021-08-09 |
| **摘 要：**  为了方便客户升级TC2中的PLC程序，可以借助功能块“FB\_CopeFileToFile”实现程序更换。 | |
| **附 件：**   |  |  |  | | --- | --- | --- | | 序 号 | 文件名 | 备注 | |  |  |  | |  |  |  | |  |  |  | |  |  |  | | |
| **历史版本：**   |  |  |  | | --- | --- | --- | | 2021-08-09 | 夏璐杰 | TC2中自动更换PLC运行程序并重启 | |  |  |  | |  |  |  | |  |  |  | |  |  |  | | |
| **免责声明：**  我们已对本文档描述的内容做测试。但是差错在所难免，无法保证绝对正确并完全满足您的使用需求。本文档的内容可能随时更新，如有改动，恕不事先通知，也欢迎您提出改进建议。 | |
| **参考信息：** | |

目 录

[1. 软硬件版本 2](#_Toc109827063)

[1.1. 倍福Beckhoff 2](#_Toc109827064)

[1.1.1. 控制器硬件 2](#_Toc109827065)

[1.1.2. 控制软件 2](#_Toc109827066)

[2. TC2中Boot文件夹介绍 2](#_Toc109827067)

[3. FB\_CopeFileToFile功能块介绍 3](#_Toc109827068)

[3.1. FB\_CopeFileToFile功能块用法 3](#_Toc109827069)

[3.2. 功能块FB\_FileCopy源代码 4](#_Toc109827070)

[4. 测试程序介绍 4](#_Toc109827071)

[4.1. 硬件环境 4](#_Toc109827072)

[4.2. 软件环境 4](#_Toc109827073)

[5. 附录 8](#_Toc109827074)

[1.1 功能块FB\_CopeFileToFile源代码 14](#_Toc109827075)

# 软硬件版本

## 倍福Beckhoff

### 控制器硬件

CX8090

EL2008

EL2004

### 控制软件

笔记本和控制器都是基于TwinCAT 2.11 Build 2307版本

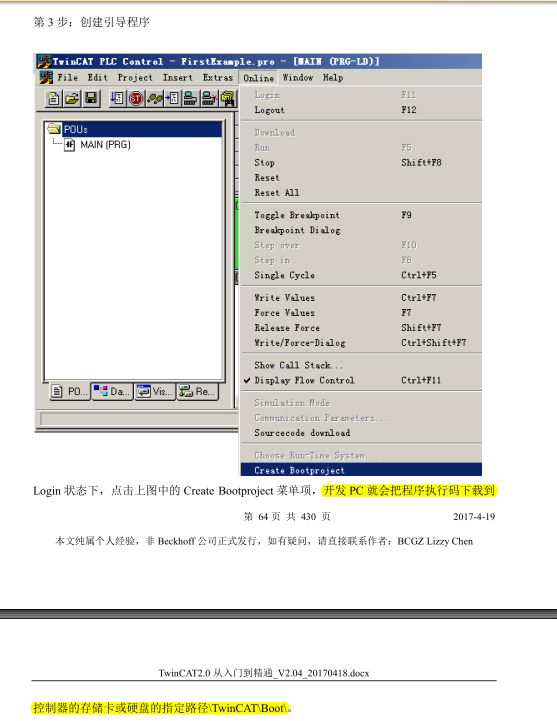
# TC2中Boot文件夹介绍

在倍福控制器中， TwinCAT 2的项目文件总是位于“hard disk:\TwinCAT\Boot”或“C:\TwinCAT\Boot”中，所以对于某些控制器无法启动的情况，可以通过清空boot文件夹来实现TwinCAT 项目恢复出厂设置，以使控制器恢复至正常状态。

对于TwinCAT2的控制器，其Boot文件夹中的内容一般如下



其中TCPLC\_P\_1.wbp为引导程序 Boot project，其为每次 TwinCAT PLC Control 中执行 Create BootProject 的时候生成。

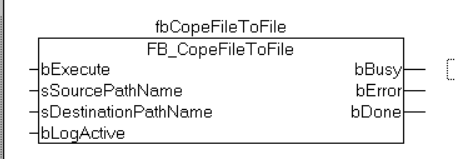


因此，对于同一套控制器及其外围设备，如果客户想进行程序升级，可以通过更换TCPLC\_P\_1.wbp来实现。

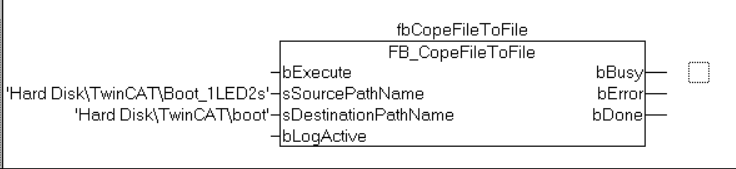
# FB\_CopeFileToFile功能块介绍

## FB\_CopeFileToFile功能块用法

本文档封装了一个自定义功能块“FB\_CopeFileToFile”，其功能是把文件夹1中的内容复制至文件夹2中。



为了实现TCPLC\_P\_1.wbp的更换升级，则可以把新程序TCPLC\_P\_1.wbp放置在hard disk:\TwinCAT\Boot\_1LED2s文件中，则当bExecute为true时，会覆盖掉hard disk:\TwinCAT\Boot中旧的TCPLC\_P\_1.wbp。



## 功能块FB\_FileCopy源代码

FUNCTION\_BLOCK FB\_FileCopy

VAR\_INPUT

srcNetId : T\_AmsNetId; (\* TwinCAT network address of the source file \*)

srcFileName : T\_MaxString; (\* source file path and name \*)

destNetId : T\_AmsNetId; (\* TwinCAT network address of the destination file \*)

destFileName : T\_MaxString; (\* destination file path and name \*)

bExecute : BOOL; (\* rising edge start fb execution \*)

tAdsTimeOut : TIME; (\* ads timeout \*)

bLogActive : BOOL:=FALSE;

END\_VAR

VAR\_OUTPUT

bBusy :BOOL;

bError :BOOL;

nErrId :UDINT;

END\_VAR

VAR

fbFileOpen :FB\_FileOpen;

fbFileClose :FB\_FileClose;

fbFileRead :FB\_FileRead;

fbFileWrite :FB\_FileWrite;

hSrcFile :UINT := 0;(\* file handle of the source file \*)

hDestFile :UINT := 0;(\* file handle of the destination file \*)

Step :DWORD;

RisingEdge :R\_TRIG;

buffRead :ARRAY[1..30000] OF BYTE;(\* buffer \*)

cbReadLength :UDINT:=0;

cbWriteActLength: UDINT:=0;

cnRWCounter: UDINT:=0;

OnceMore: BOOL;

cnRetryCounter: UDINT;

END\_VAR

VAR CONSTANT

FileIsEmpty: INT:=16#1234;

END\_VAR

RisingEdge(CLK:=bExecute);

IF RisingEdge.Q THEN

IF NOT bBusy THEN

bBusy := TRUE;

bError:= FALSE;

nErrId:=0;

Step := 1;

cbReadLength:=0;

hSrcFile:=0;

hDestFile:=0;

cnRWCounter:=0;

cnRetryCounter:=0;

END\_IF

END\_IF

CASE Step OF

0: (\* do nothing \*)

;

1: (\* open source file \*)

OnceMore:=FALSE;

cnRWCounter:=0;

fbFileOpen( bExecute := FALSE );

fbFileOpen( sNetId := srcNetId,

sPathName := srcFileName,

nMode := FOPEN\_MODEREAD OR FOPEN\_MODEBINARY,

ePath := PATH\_GENERIC,

tTimeout := tAdsTimeout,

bExecute := TRUE );

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='open source file,step %d' , dintArg := Step);END\_IF

Step := Step + 1;

2:

fbFileOpen( bExecute := FALSE );

IF ( NOT fbFileOpen.bBusy ) THEN

IF ( fbFileOpen.bError ) THEN

nErrId := fbFileOpen.nErrId;

bError := TRUE;

Step := 50;

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:=CONCAT(CONCAT(srcFileName,destFileName),'File Open Error,ErrorId %d') , dintArg := nErrId);END\_IF

ELSE

hSrcFile := fbFileOpen.hFile;

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='get source file handle ,step %d' , dintArg := Step);END\_IF

Step := Step + 1;

END\_IF

END\_IF

3: (\* open destination file \*)

fbFileOpen( bExecute := FALSE );

fbFileOpen( sNetId := destNetId,

sPathName := destFileName,

nMode := FOPEN\_MODEWRITE OR FOPEN\_MODEBINARY,

ePath := PATH\_GENERIC,

tTimeout := tAdsTimeout,

bExecute := TRUE );

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='open destination file,step %d' , dintArg := Step);END\_IF

Step := Step+1;

4:

fbFileOpen( bExecute := FALSE );

IF ( NOT fbFileOpen.bBusy ) THEN

IF ( fbFileOpen.bError ) THEN

nErrId := fbFileOpen.nErrId;

bError := TRUE;

Step := 50;

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:=CONCAT(CONCAT(srcFileName,destFileName),'File Open Error,ErrorId %d') , dintArg := nErrId);END\_IF

ELSE

hDestFile := fbFileOpen.hFile;

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='get destination file handle ,step %d' , dintArg := Step);END\_IF

Step := Step + 1;

END\_IF

END\_IF

5: (\*read data from source file\*)

cbReadLength := 0;

fbFileRead( bExecute:= FALSE );

fbFileRead( sNetId:=SrcNetId,

hFile:=hSrcFile,

pReadBuff:= ADR(buffRead),

cbReadLen:= SIZEOF(buffRead),

bExecute:=TRUE,

tTimeout:=tAdsTimeOut );

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='read data from source file,step %d' , dintArg := Step);END\_IF

Step := Step + 1;

6:

fbFileRead( bExecute:= FALSE );

IF ( NOT fbFileRead.bBusy ) THEN

(\*//fbFileRead.bEOF;\*)

IF ( fbFileRead.bError ) THEN

nErrId := fbFileRead.nErrId;

bError := TRUE;

Step := 50;

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:=CONCAT(CONCAT(srcFileName,destFileName),'File Read Error,ErrorId %d') , dintArg := nErrId);END\_IF

ELSE

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='get Read Size ,step %d' , dintArg := Step);END\_IF

cbReadLength := fbFileRead.cbRead;

Step := Step + 1;

END\_IF

END\_IF

7: (\*write data to destination file\*)

fbFileWrite( bExecute := FALSE );

fbFileWrite( sNetId:=DestNetId,

hFile:=hDestFile,

pWriteBuff:= ADR(buffRead),

cbWriteLen:= cbReadLength,

bExecute:=TRUE,

tTimeout:=tAdsTimeOut );

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='write data to destination file ,step %d' , dintArg := Step);END\_IF

Step := Step + 1;

cnRWCounter:=cnRWCounter+1;

8:

fbFileWrite( bExecute := FALSE );

IF ( NOT fbFileWrite.bBusy ) THEN

cbWriteActLength:=fbFileWrite.cbWrite;

IF ( fbFileWrite.bError ) THEN

nErrId := fbFileWrite.nErrId;

bError := TRUE;

Step := 50;

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:=CONCAT(CONCAT(srcFileName,destFileName),'File Write Error,ErrorId %d') , dintArg := nErrId);END\_IF

ELSIF (cnRWCounter=1) AND (cbWriteActLength=0) THEN

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_ERROR , msgFmtStr:=CONCAT(CONCAT(CONCAT(CONCAT('Read:',UDINT\_TO\_STRING(cbReadLength)),' write:'),UDINT\_TO\_STRING(cbWriteActLength)),'\_step %d') , dintArg := Step);END\_IF

OnceMore:=TRUE;

cnRetryCounter:=cnRetryCounter+1;

IF cnRetryCounter>10 THEN

bError := TRUE;

OnceMore:=FALSE;

nErrId := FileIsEmpty;

END\_IF

Step := 50; (\* repeat reading \*)

ELSE

IF fbFileRead.bEOF THEN (\*Check if the EOF flag ist set \*)

Step := 50; (\*cleanup: close the destination and source files \*)

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='Check if the EOF flag ist set ,step %d' , dintArg := Step);END\_IF

ELSE

Step := 5; (\* repeat reading \*)

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='repeat reading ,step %d' , dintArg := Step);END\_IF

END\_IF

END\_IF

END\_IF

30: (\* close the destination file \*)

fbFileClose( bExecute := FALSE );

fbFileClose( sNetId:=destNetId,

hFile:=hDestFile,

bExecute:=TRUE,

tTimeout:=tAdsTimeOut );

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='close the destination file ,step %d' , dintArg := Step);END\_IF

Step := Step + 1;

31:

fbFileClose( bExecute := FALSE );

IF ( NOT fbFileClose.bBusy ) THEN

IF ( fbFileClose.bError ) THEN

nErrId := fbFileClose.nErrId;

bError := TRUE;

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:=CONCAT(CONCAT(srcFileName,destFileName),'File Close Error,ErrorId %d') , dintArg := nErrId);END\_IF

END\_IF

Step := 50;

hDestFile := 0;

END\_IF

40: (\* close source file \*)

fbFileClose( bExecute := FALSE );

fbFileClose( sNetId:=srcNetId,

hFile:=hSrcFile,

bExecute:=TRUE,

tTimeout:=tAdsTimeOut );

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='close source file ,step %d' , dintArg := Step);END\_IF

Step := Step + 1;

41:

fbFileClose( bExecute := FALSE );

IF (NOT fbFileClose.bBusy ) THEN

IF ( fbFileClose.bError ) THEN

nErrId := fbFileClose.nErrId;

bError := TRUE;

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:=CONCAT(CONCAT(srcFileName,destFileName),'File Close Error,ErrorId %d') , dintArg := nErrId);END\_IF

END\_IF

Step := 50;

hSrcFile := 0;

END\_IF

50: (\*error or ready => cleanup\*)

IF ( hDestFile <> 0 ) THEN

Step := 30; (\* close the destination file\*)

ELSIF (hSrcFile <> 0 ) THEN

Step := 40; (\* close the source file \*)

ELSE

IF OnceMore THEN

Step:=1;

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='OnceMore! ,step %d' , dintArg := Step);END\_IF

ELSE

Step := 0; (\* ready \*)

IF bLogActive THEN ADSLOGDINT(msgCtrlMask:=ADSLOG\_MSGTYPE\_LOG , msgFmtStr:='Done! ,step %d' , dintArg := Step);END\_IF

bBusy := FALSE;

END\_IF

END\_IF

END\_CASE

## 功能块FB\_CopeFileToFile源代码

FUNCTION\_BLOCK FB\_CopeFileToFile

VAR\_INPUT

bExecute: BOOL;

sSourcePathName: STRING:='Hard Disk\TwinCAT\Boot\_4LED1s';

sDestinationPathName: STRING:='Hard Disk\TwinCAT\boot';

bLogActive: BOOL:=FALSE;

END\_VAR

VAR\_OUTPUT

bBusy: BOOL;

bError: BOOL;

bDone: BOOL;

END\_VAR

VAR

nState: INT;

fbCreateDir: FB\_CreateDir;

bOldExecute: BOOL;

nCount: INT;

nCount2: INT;

fileList: ARRAY [1..cnMaxNoOfFiles] OF ST\_FindFileEntry;

fbFindFileList: FB\_EnumFindFileList;

fbCopyFiles: FB\_FileCopy;

TonDelay: TON;

END\_VAR

VAR CONSTANT

cnMaxNoOfFiles: INT:=50;

END\_VAR

CASE nState OF

0:

IF bExecute AND NOT bBusy THEN

IF NOT bOldExecute THEN

bBusy := TRUE;

bError := FALSE;

bDone := FALSE;

nState := 30;

nCount:=1;

nCount2:=1;

END\_IF

END\_IF

30: (\* Update file list \*)

IF nCount <= cnMaxNoOfFiles THEN

fbFindFileList

(

sNetID:='',

sPathName:=CONCAT(sSourcePathName,'\\*.\*'),

eCmd:= eEnumCmd\_First,

pFindList:= ADR(fileList),

cbFindList:=SIZEOF(fileList),

tTimeout:=t#10s,

bExecute := TRUE

);

IF fbFindFileList.bExecute AND NOT fbFindFileList.bBusy THEN

fbFindFileList(bExecute := FALSE);

fbCopyFiles(srcNetId:='',destNetId:='', tAdsTimeOut:=t#10s);

nCount2:=1;

nState:=40;

END\_IF

IF fbFindFileList.bError THEN

nState := 0;

fbFindFileList(bExecute := FALSE);

bBusy := FALSE;

bError := TRUE;

END\_IF

END\_IF

40:

(\* copy each file from source to destination folder \*)

IF fileList[nCount2].sFileName <> '' THEN

fbCopyFiles.srcFileName := CONCAT(CONCAT(sSourcePathName,'\'),fileList[nCount2].sFileName);

fbCopyFiles.destFileName := CONCAT(CONCAT(sDestinationPathName,'\'),fileList[nCount2].sFileName);

fbCopyFiles(bExecute := TRUE);

IF fbCopyFiles.bExecute AND NOT fbCopyFiles.bBusy THEN

fbCopyFiles(bExecute := FALSE);

nCount2 := nCount2+1;

END\_IF

IF fbCopyFiles.bError THEN

bBusy := FALSE;

bError := TRUE;

nState := 0;

END\_IF

ELSE

nCount:=nCount+1;

nState:=50;

END\_IF

50:

TonDelay(in:=TRUE,pt:=t#1s);

IF TonDelay.Q THEN

TonDelay(in:=FALSE,pt:=t#1s);

nState:=0;

bDone:=TRUE;

bBusy := FALSE;

END\_IF

END\_CASE

# 测试程序介绍

## 硬件环境

硬件配置为CX8090+EL2008+EL1004，其中EL1004的1-4号针脚关联变量bIn[1..4]，EL2008的1-4号针脚关联变量bIed[1..4]。

图形用户界面, 文本, 应用程序

描述已自动生成

## 软件环境

CX8090中原始程序如下，加载了Utilities库。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

现更换程序，把新程序下载到控制器，并create bootproject，然后从boot文件夹中复制TCPLC\_P\_1到Boot\_4LED1s。

图形用户界面, 文本

描述已自动生成

再更换程序，把新程序下载到控制器，并create bootproject，然后从boot文件夹中复制TCPLC\_P\_1到Boot\_4LED1s。

图形用户界面, 文本

描述已自动生成

分析ACT\_changAndReboot程序可知：

文本

描述已自动生成

当bIn[3]为true时，nSelect:=1，则新程序的目录为Hard Disk\TwinCAT\Boot\_4LED1s，若bStart为true，则换更换新程序并重启。

图形用户界面

描述已自动生成

当bIn[3]为false时，nSelect:=2，则新程序的目录为Hard Disk\TwinCAT\Boot\_1LED2s，若bStart为true，则换更换新程序并重启。

图形用户界面

描述已自动生成

**上海（ 中国区总部）**

中国上海市静安区汶水路 299 弄 9号（市北智汇园）

电话: 021-66312666

**北京分公司**

北京市西城区新街口北大街 3 号新街高和大厦 407 室

电话: 010-82200036 邮箱: beijing@beckhoff.com.cn

**广州分公司**

广州市天河区珠江新城珠江东路32号利通广场1303室

电话: 020-38010300/1/2 邮箱: guangzhou@beckhoff.com.cn

**成都分公司**

成都市锦江区东御街18号 百扬大厦2305 室

电话: 028-86202581 邮箱: chengdu@beckhoff.com.cn

|  |  |
| --- | --- |
| 请用微信扫描二维码  通过公众号与技术支持交流 | 倍福官方网站：  https://www.beckhoff.com.cn  在线帮助系统：  https://infosys.beckhoff.com/index\_en.htm |
| 倍福虚拟学院：  https://tr.beckhoff.com.cn/ |
| 招贤纳士：job@beckhoff.com.cn  技术支持：support@beckhoff.com.cn  产品维修：service@beckhoff.com.cn  方案咨询：sales@beckhoff.com.cn |
|  |