CML Microcircuits

COMMUNICATION SEMICONDUCTORS

EC0003 C Driver Libraries CML_UtilsLib

Publication: CL/EC0003/UM/2 February 2016

EC0003 C Code Development Environment
GUI Interface Library

Contents

1 Introduction 4
1.1 History 4
2 File Index 5
2.1 File List 5
3 Class Documentation 6
3.1 CML_FUNCTIONIMAGE Struct Reference 6
3.1.1 Public Attributes 6
3.1.2 Detailed Description 6
3.1.3 Member Data Documentation 6
3.2 CML_FUNCTIONIMAGEBLOCK Struct Reference 7
3.2.1 Public Attributes 7
3.2.2 Detailed Description 7
3.2.3 Member Data Documentation 7
4 File Documentation 8
4.1 inc/fi_tools.h File Reference 8
4.1.1 Classes 8
4.1.2 *Stores an FI.* Macros 8
4.1.3 Functions 8
4.1.4 Macro Definition Documentation 8
4.1.5 Function Documentation 9
4.2 inc/messages.h File Reference 13
4.2.1 Macros 13
4.2.2 Macro Definition Documentation 14
4.3 inc/mon_dialog.h File Reference 18
4.3.1 Macros 18
4.3.2 Functions 18
4.3.3 Macro Definition Documentation 18
4.3.4 Function Documentation 19
4.4 inc/mon_environment.h File Reference 21
4.4.1 Macros 21
4.4.2 Functions 21
4.4.3 Macro Definition Documentation 21
4.4.4 Function Documentation 21

4.5 inc/mon_fi.h File Reference 23
4.5.1 Macros 23
4.5.2 Functions 23
4.5.3 Macro Definition Documentation 24
4.6 inc/mon_file.h File Reference 33
4.6.1 Macros 33
4.6.2 Functions 33
4.6.3 Macro Definition Documentation 35
4.6.4 Function Documentation 37
4.7 inc/mon_general.h File Reference 47
4.7.1 Functions 47
4.7.2 Function Documentation 47
4.8 inc/monitor.h File Reference 50
4.8.1 Macros 50
4.8.2 Functions 50
4.8.3 Macro Definition Documentation 50
4.8.4 Function Documentation 50

1 Introduction

This document provides a description of the CML_UtilsLib. This library is used by the EC0003 GUI which provides an interface for human interaction with the PE0003. The document has been created using Doxygen – an automatic documentation generation tool used to produce software reference documents. Content is created from within the code itself and therefore offers intuitive cross referencing between the document and code and provides an easy path to future updating.

1.1 History

Version	Changes	Date
1	First Release	22 July 2015
2	Section 4.6: Updated mon_file.h Added functions to manage wav files	08 February 2016

2 File Index

2.1 File List

The following is a list of all files. Brief descriptions are given for each in the relevant sections.

inc/fi_tools.h
inc/messages.h
inc/mon_control.h
inc/mon_dialog.h
inc/mon_environment.h
inc/mon_fi.h
inc/mon_file.h
inc/mon_general.h
inc/monitor.h
src/fi_tools.c
src/mon_control.c
src/mon_dialog.c
src/mon_environment.c
src/mon_fi.c
src/mon_file.c
src/mon_general.c
src/monitor.c

3 Class Documentation

3.1 CML_FUNCTIONIMAGE Struct Reference

Stores an FI.

```
#include <fi_tools.h>
```

3.1.1 Public Attributes

- uint8_t **nBlocks**
  Number of blocks.
- struct **CML_FUNCTIONIMAGEBLOCK** ** **block**
  Pointer to the block array.
- uint16_t **activatePtr**
  FI start address.
- uint16_t **activateLen**
  Lenght of start block =0.

3.1.2 Detailed Description

Stores an FI.

3.1.3 Member Data Documentation

uint16_t CML_FUNCTIONIMAGE::activateLen

Lenght of start block =0.

uint16_t CML_FUNCTIONIMAGE::activatePtr

FI start address.

struct CML_FUNCTIONIMAGEBLOCK** CML_FUNCTIONIMAGE::block

Pointer to the block array.

uint8_t CML_FUNCTIONIMAGE::nBlocks

Number of blocks.

The documentation for this struct was generated from the following file:

- inc/**fi_tools.h**

3.2 CML_FUNCTIONIMAGEBLOCK Struct Reference

Stores a block of a function image.

```
#include <fi_tools.h>
```

3.2.1 Public Attributes

- uint16_t **dbPtr**
  Pointer in memory in the device.
- uint16_t **dbLen**
  Length of block in words.
- uint16_t **dbChkHi**
  Checksum high.
- uint16_t **dbChkLo**
  Checksum low.
- uint16_t * **db**
  Pointer to the data array.

3.2.2 Detailed Description

Stores a block of a function image.

3.2.3 Member Data Documentation

uint16_t* CML_FUNCTIONIMAGEBLOCK::db

Pointer to the data array.

uint16_t CML_FUNCTIONIMAGEBLOCK::dbChkHi

Checksum high.

uint16_t CML_FUNCTIONIMAGEBLOCK::dbChkLo

Checksum low.

uint16_t CML_FUNCTIONIMAGEBLOCK::dbLen

Length of block in words.

uint16_t CML_FUNCTIONIMAGEBLOCK::dbPtr

Pointer in memory in the device.

- **??**

4 File Documentation

4.1 inc/fi_tools.h File Reference

```
#include "lpc_types.h"
#include "chip.h"
```

4.1.1 Classes

- struct **CML_FUNCTIONIMAGEBLOCK**
- *Stores a block of a function image.* struct **CML_FUNCTIONIMAGE**

4.1.2 *Stores an FI.* Macros

- #define **CML_TARGET_RAM** 0x0001
- #define **CML_TARGET_MEM** 0xABCD
- #define **CML_RAM_TIME** 300000
- #define **FI_OK** 0
  OK.
- #define **FI_ERROR** 1
  General error.
- #define **FI_WR_CHECKSUM** 2
  Wrong checksum.
- #define **FI_WR_ACT_CODE** 3
  Wrong activation code.

4.1.3 Functions

- uint8_t **Fi_LoadFunctionImageCbus_CMX7x3x** (LPC_SSP_T *CBusPort, struct **CML_FUNCTIONIMAGE** *fi, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI stored in memory to a CMX7x3x device.
- uint8_t **Fi_LoadFunctionImageCbus_CMX704x** (LPC_SSP_T *CBusPort, struct **CML_FUNCTIONIMAGE** *fi, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI stored in memory to a CMX704x device.
- uint8_t **Fi_LoadFunctionImageCbus_CMX714x** (LPC_SSP_T *CBusPort, struct **CML_FUNCTIONIMAGE** *fi, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI stored in memory to a CMX714x device.
- uint8_t **Fi_LoadFunctionImageCbus_CMX724x** (LPC_SSP_T *CBusPort, struct **CML_FUNCTIONIMAGE** *fi, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI stored in memory to a CMX724x device.
- uint8_t **Fi_LoadFunctionImageCbus_CMX734x** (LPC_SSP_T *CBusPort, struct **CML_FUNCTIONIMAGE** *fi, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI stored in memory to a CMX734x device.
- uint8_t **Fi_LoadFunctionImageCbus_CMX7x6x** (LPC_SSP_T *CBusPort, struct **CML_FUNCTIONIMAGE** *fi, uint16_t *productId, uint16_t *fiVersion)
  Load a FI stored in memory to a CMX7x6x device.

4.1.4 Macro Definition Documentation

#define CML_RAM_TIME 300000

#define CML_TARGET_MEM 0xABCD

#define CML_TARGET_RAM 0x0001

#define FI_ERROR 1

General error.

#define FI_OK 0

OK.

#define FI_WR_ACT_CODE 3

Wrong activation code.

#define FI_WR_CHECKSUM 2

Wrong checksum.

4.1.5 Function Documentation

uint8_t Fi_LoadFunctionImageCbus_CMX704x (LPC_SSP_T * *CBusPort*, struct CML_FUNCTIONIMAGE * *fi*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI stored in memory to a CMX704x device.

Parameters:

CBusPort	- CBUS1, CBUS2
fi	- Fi struct that stores the whole FI
dwActivationCode	- Activation code required by the FI
productId	- Pointer that returns the product ID
fiVersion	- Pointer that returns the FI version

Returns:

result - Returns FI_OK, FI_ERROR, FI_WR_CHECKSUM, FI_WR_ACT_CODE,

Note:

Uses the **CML_FUNCTIONIMAGE** structure to store the FI

uint8_t Fi_LoadFunctionImageCbus_CMX714x (LPC_SSP_T * *CBusPort*, struct CML_FUNCTIONIMAGE * *fi*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI stored in memory to a CMX714x device.

Parameters:

CBusPort	- CBUS1, CBUS2
fi	- Fi struct that stores the whole FI
dwActivationCode	- Activation code required by the FI
productId	- Pointer that returns the product ID
fiVersion	- Pointer that returns the FI version

Returns:

result - Returns FI_OK, FI_ERROR, FI_WR_CHECKSUM, FI_WR_ACT_CODE,

Note:

Uses the **CML_FUNCTIONIMAGE** structure to store the FI

uint8_t Fi_LoadFunctionImageCbus_CMX724x (LPC_SSP_T * *CBusPort*, struct CML_FUNCTIONIMAGE * *fi*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI stored in memory to a CMX724x device.

Parameters:

CBusPort	- CBUS1, CBUS2
fi	- Fi struct that stores the whole FI
dwActivationCode	- Activation code required by the FI
productId	- Pointer that returns the product ID
fiVersion	- Pointer that returns the FI version

Returns:

result - Returns FI_OK, FI_ERROR, FI_WR_CHECKSUM, FI_WR_ACT_CODE,

Note:

Uses the **CML_FUNCTIONIMAGE** structure to store the FI

uint8_t Fi_LoadFunctionImageCbus_CMX734x (LPC_SSP_T * *CBusPort*, struct CML_FUNCTIONIMAGE * *fi*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI stored in memory to a CMX734x device.

Parameters:

CBusPort	- CBUS1, CBUS2

fi	- Fi struct that stores the whole FI
dwActivationCode	- Activation code required by the FI
productId	- Pointer that returns the product ID
fiVersion	- Pointer that returns the FI version

Returns:

result - Returns FI_OK, FI_ERROR, FI_WR_CHECKSUM, FI_WR_ACT_CODE,

Note:

Uses the **CML_FUNCTIONIMAGE** structure to store the FI

uint8_t Fi_LoadFunctionImageCbus_CMX7x3x (LPC_SSP_T * *CBusPort*, struct CML_FUNCTIONIMAGE * *fi*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI stored in memory to a CMX7x3x device.

This library requires the following peripherals are enabled

INIT TIMER

INIT CBUS

INIT GPIO DI IO

Parameters:

CBusPort	- CBUS1, CBUS2
fi	- Fi struct that stores the whole FI
dwActivationCode	- Activation code required by the FI
productId	- Pointer that returns the product ID
fiVersion	- Pointer to returns the FI version

Returns:

result - Returns FI_OK, FI_ERROR, FI_WR_CHECKSUM, FI_WR_ACT_CODE,

Note:

Uses the **CML_FUNCTIONIMAGE** structure to store the FI

uint8_t Fi_LoadFunctionImageCbus_CMX7x6x (LPC_SSP_T * *CBusPort*, struct CML_FUNCTIONIMAGE * *fi*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI stored in memory to a CMX7x6x device.

Parameters:

CBusPort	- CBUS1, CBUS2
fi	- Fi struct that stores the whole FI

productId	- Pointer that returns the product ID
fiVersion	- Pointer that returns the FI version

Returns:

result - Returns FI_OK, FI_ERROR, FI_WR_CHECKSUM,

Note:

Uses the **CML_FUNCTIONIMAGE** structure to store the FI

4.2 inc/messages.h File Reference

4.2.1 Macros

- #define **PE0003TOPC_ASK_LINK** 0xE3
- #define **PCTOPE0003_ACK_LINK** 0x1E
- #define **PE0003_LINKED** 0x31
- #define **LEN_HEADER_FIELD** 1
- #define **LEN_MSG_FIELD** 1
- #define **LEN_DATALENGTH_FIELD** 4
- #define **LEN_DATA_FIELD VAR_LEN**
- #define **VAR_LEN** 0xFFFF
  Variable length, maximum.
- #define **EXT_LEN** 0xFFFE
  Extended length from word to dword.
- #define **MAX_LEN** 0xFFFFFFFF
  The longest message possible.
- #define **HEADER_CHAR** 0xFA
  Normal Header.
- #define **HEADER_CHAR_ASYNC** 0xFB
  Asynchronous Header (commands from the GUI without response)
- #define **HEADER_CHAR_LEN** 1
- #define **HOST_MSG_ERROR** 0x01
  Report an error //No dataframe.
- #define **HOST_MSG_CLEARDISP** 0x02
  Clear the display //No dataframe.
- #define **HOST_MSG_PRINTF** 0x03
  Output some data to PC.
- #define **HOST_MSG_SCANF** 0x04
  Input data from PC.
- #define **HOST_MSG_SCANFNONBLOCK** 0x05
  Non blocking Scanf.
- #define **HOST_MSG_ENABLENONBLOCK** 0x06
  Enables the non block scanf operation //No dataframe.
- #define **HOST_MSG_DISABLENONBLOCK** 0x07
  Disables the non block scanf opeartion //No dataframe.
- #define **HOST_MSG_DIALOG** 0x08
  Displays some dialog.
- #define **HOST_MSG_MONITOR** 0x09
  Monitor variables and controls.
- #define **HOST_MSG_ENVIRONMENT** 0x0A
  Environment variable control.
- #define **HOST_MSG_FILE** 0x0B
  File operations //No dataframe.
- #define **HOST_MSG_FI** 0x0C
  Loads an FI from GUI.
- #define **HOST_MSG_ENDPROGRAM** 0x0D
  Ends the program.
- #define **HOST_LAST_COMMAND** 0x0E
  Point last command from PC TO GUI.
- #define **MSG_OK** 0x01
  Message answer OK.

- #define **MSG_ERROR** 0x00
  Message answer ERROR.
- #define **MSG_NODATA** 0x02
  Message answer no data.
- #define **GUI_MSG_STOP** 0x01
- #define **GUI_MSG_SCANF_ANSWER** 0x02
- #define **GUI_MSG_SCANFNONBLOCK_ANSWER** 0x03
- #define **GUI_MSG_DIALOG_ANSWER** 0x04
- #define **GUI_MSG_ENVIRONMENT_ANSWER** 0x05
- #define **GUI_MSG_FILE_ANSWER** 0x06
- #define **GUI_MSG_FI_ANSWER** 0x07
- #define **GUI_MSG_RESET** 0x08
- #define **GUI_MSG_SCANF_END** 0x09
- #define **GUI_LAST_COMMAND** 0x0A
- #define **M_BOOL** 0x01
- #define **M_INT** 0x02
- #define **M_SHORT** 0x03
- #define **M_BYTE** 0x04
- #define **M_CHAR** 0x05
- #define **M_STRING** 0x06
- #define **M_FLOAT** 0x07

4.2.2 Macro Definition Documentation

#define EXT_LEN 0xFFFE

Extended length from word to dword.

#define GUI_LAST_COMMAND 0x0A

#define GUI_MSG_DIALOG_ANSWER 0x04

#define GUI_MSG_ENVIRONMENT_ANSWER 0x05

#define GUI_MSG_FI_ANSWER 0x07

#define GUI_MSG_FILE_ANSWER 0x06

#define GUI_MSG_RESET 0x08

#define GUI_MSG_SCANF_ANSWER 0x02

#define GUI_MSG_SCANF_END 0x09

#define GUI_MSG_SCANFNONBLOCK_ANSWER 0x03

#define GUI_MSG_STOP 0x01

#define HEADER_CHAR 0xFA

Normal Header.

#define HEADER_CHAR_ASYNC 0xFB

Asynchronous Header (commands from the GUI without response)

#define HEADER_CHAR_LEN 1

#define HOST_LAST_COMMAND 0x0E

Point last command from PC TO GUI.

#define HOST_MSG_CLEARDISP 0x02

Clear the display //No dataframe.

#define HOST_MSG_DIALOG 0x08

Displays some dialog.

#define HOST_MSG_DISABLENONBLOCK 0x07

Disables the non block scanf opeartion //No dataframe.

#define HOST_MSG_ENABLENONBLOCK 0x06

Enables the non block scanf operation //No dataframe.

#define HOST_MSG_ENDPROGRAM 0x0D

Ends the program.

#define HOST_MSG_ENVIRONMENT 0x0A

Environment variable control.

#define HOST_MSG_ERROR 0x01

Report an error //No dataframe.

#define HOST_MSG_FI 0x0C

Loads an FI from GUI.

#define HOST_MSG_FILE 0x0B

File operations //No dataframe.

#define HOST_MSG_MONITOR 0x09

Monitor variables and controls.

#define HOST_MSG_PRINTF 0x03

Output some data to PC.

#define HOST_MSG_SCANF 0x04

Input data from PC.

#define HOST_MSG_SCANFNONBLOCK 0x05

Non blocking Scanf.

#define LEN_DATA_FIELD VAR_LEN

#define LEN_DATALENGTH_FIELD 4

#define LEN_HEADER_FIELD 1

#define LEN_MSG_FIELD 1

#define M_BOOL 0x01

#define M_BYTE 0x04

#define M_CHAR 0x05

#define M_FLOAT 0x07

#define M_INT 0x02

#define M_SHORT 0x03

#define M_STRING 0x06

#define MAX_LEN 0xFFFFFFFF

The longest message possible.

#define MSG_ERROR 0x00

Message answer ERROR.

#define MSG_NODATA 0x02

Message answer no data.

#define MSG_OK 0x01

Message answer OK.

#define PCTOPE0003_ACK_LINK 0x1E

#define PE0003_LINKED 0x31

#define PE0003TOPC_ASK_LINK 0xE3

#define VAR_LEN 0xFFFF

Variable length, maximum.

4.3 inc/mon_dialog.h File Reference

4.3.1 Macros

- #define **HOST_DIALOG_MESSAGE** 0x01
- #define **HOST_DIALOG_YESNO** 0x02
- #define **HOST_DIALOG_ENTRY** 0x03
- #define **HOST_DIALOGTYPE_NONE** 0x01
- #define **HOST_DIALOGTYPE_INFO** 0x02
- #define **HOST_DIALOGTYPE_WARNING** 0x03
- #define **HOST_DIALOGTYPE_ERROR** 0x04
- #define **GUI_DIALOG_MESSAGE** 0x01
- #define **GUI_DIALOG_YESNO** 0x02
- #define **GUI_DIALOG_ENTRY** 0x03
- #define **GUI_DIALOG_RESULT_ERROR** 0xFE
- #define **GUI_DIALOG_RESULT_OK** 0x01
- #define **GUI_DIALOG_RESULT_CANCEL** 0x02
- #define **GUI_DIALOG_RESULT_YES** 0x03
- #define **GUI_DIALOG_RESULT_NO** 0x04

4.3.2 Functions

- uint8_t **Gui_DialogYesNo** (const char *str)
  Displays a Yes/No dialog.
- uint8_t **Gui_DialogTitleYesNo** (const char *str, const char *title)
  Displays a Yes/No dialog with a title.
- uint8_t **Gui_DialogMessage** (const char *str, const char *title, uint8_t typeDialogMsg)
  Displays a message window.
- uint8_t **Gui_DialogInfo** (const char *str)
  Displays a message window.

4.3.3 Macro Definition Documentation

#define GUI_DIALOG_ENTRY 0x03

#define GUI_DIALOG_MESSAGE 0x01

#define GUI_DIALOG_RESULT_CANCEL 0x02

#define GUI_DIALOG_RESULT_ERROR 0xFE

#define GUI_DIALOG_RESULT_NO 0x04

#define GUI_DIALOG_RESULT_OK 0x01

#define GUI_DIALOG_RESULT_YES 0x03

#define GUI_DIALOG_YESNO 0x02

#define HOST_DIALOG_ENTRY 0x03

#define HOST_DIALOG_MESSAGE 0x01

#define HOST_DIALOG_YESNO 0x02

#define HOST_DIALOGTYPE_ERROR 0x04

#define HOST_DIALOGTYPE_INFO 0x02

#define HOST_DIALOGTYPE_NONE 0x01

#define HOST_DIALOGTYPE_WARNING 0x03

4.3.4 Function Documentation

uint8_t Gui_DialogInfo (const char * *str*)

Displays a message window.

Parameters:

str	- question

Returns:

result - GUI_DIALOG_RESULT_ERROR
GUI_DIALOG_RESULT_OK
GUI_DIALOG_RESULT_CANCEL

uint8_t Gui_DialogMessage (const char * *str*, const char * *title*, uint8_t *typeDialogMsg*)

Displays a message window.

Parameters:

str	- question
title	- title of the dialog window
typeDialogMsg	- display an dialog type icon HOST_DIALOGTYPE_NONE HOST_DIALOGTYPE_INFO HOST_DIALOGTYPE_WARNING HOST_DIALOGTYPE_ERROR

Returns:

result - GUI_DIALOG_RESULT_ERROR
GUI_DIALOG_RESULT_OK
GUI_DIALOG_RESULT_CANCEL

```
uint8_t res;
res = Gui_DialogMessage("Error produced during execution", "PE00003 ERROR",
HOST_DIALOGTYPE_ERROR);
                 if (res ==  GUI_DIALOG_RESULT_CANCEL)
                     ...
```

uint8_t Gui_DialogTitleYesNo (const char * *str*, const char * *title*)

Displays a Yes/No dialog with a title.

Parameters:

str	- question
title	- title of the dialog window

Returns:

result - GUI_DIALOG_RESULT_ERROR
GUI_DIALOG_RESULT_YES
GUI_DIALOG_RESULT_NO

```
uint8_t res;

res = Gui_DialogTitleYesNo("Do you want to continue?", "PE0003 QUESTION!!!");
if (res ==  GUI_DIALOG_RESULT_YES)
     ...
```

uint8_t Gui_DialogYesNo (const char * *str*)

Displays a Yes/No dialog.

Parameters:

str	- question

Returns:

result - GUI_DIALOG_RESULT_ERROR
GUI_DIALOG_RESULT_YES
GUI_DIALOG_RESULT_NO

```
uint8_t res;

res = Gui_DialogYesNo("Do you want to continue?");
if (res ==  GUI_DIALOG_RESULT_YES)
     ...
```

4.4 inc/mon_environment.h File Reference

4.4.1 Macros

- #define **HOST_ENV_GETVAR** 0x01
- #define **HOST_ENV_SETVAR** 0x02
- #define **GUI_ENV_RETURNVAR** 0x40
- #define **HOST_ENVTYPE_ESTRING** 0x11
- #define **HOST_ENVTYPE_EINT** 0x12
- #define **ENV_NOT_FOUND** 0x20
- #define **ENV_FOUND** 0x21

4.4.2 Functions

- uint8_t **Gui_GetEnvVar** (const char *str, uint32_t *value, uint8_t type)
  Get the value of an environment variable.

4.4.3 Macro Definition Documentation

#define ENV_FOUND 0x21

#define ENV_NOT_FOUND 0x20

#define GUI_ENV_RETURNVAR 0x40

#define HOST_ENV_GETVAR 0x01

#define HOST_ENV_SETVAR 0x02

#define HOST_ENVTYPE_EINT 0x12

#define HOST_ENVTYPE_ESTRING 0x11

4.4.4 Function Documentation

uint8_t Gui_GetEnvVar (const char * *str*, uint32_t * *value*, uint8_t *type*)

Get the value of an environment variable.

Parameters:

str	- Name of the environment variable to read
value	- Pointer that returns the value of the environment variable
type	- Type of the variable HOST_ENVTYPE_ESTRING, HOST_ENVTYPE_EINT

Returns:

result - TRUE - environment variable found, FALSE environment variable not found

Note:

Environment variables are set in the GUI using a file that can be stored and loaded

4.5 inc/mon_fi.h File Reference

```
#include "lpc_types.h"
#include "chip.h"
```

4.5.1 Macros

- #define **CML_TARGET_RAM** 0x0001
- #define **CML_TARGET_MEM** 0xABCD
- #define **CML_RAM_TIME** 300000
- #define **FI_OK** 0
  OK.
- #define **FI_ERROR** 1
  General error.
- #define **FI_WR_CHECKSUM** 2
  Wrong checksum.
- #define **FI_WR_ACT_CODE** 3
  Wrong activation code.
- #define **FI_ERROR2** 4
  Error, might be wrong FI file path.
- #define **HOST_FI_LOAD_CMX7X3X** 0x01
- #define **HOST_FI_LOAD_CMX704X_CMX714X** 0x02
- #define **HOST_FI_LOAD_CMX724X_CMX734X** 0x03
- #define **HOST_FI_LOAD_CMX7X6X** 0x04
- #define **HOST_FI_STATUS** 0x05
- #define **HOST_FI_PRODUCTID** 0x06
- #define **HOST_FI_FIVERSION** 0x07
- #define **HOST_FI_SENDBLOCK** 0x08
- #define **HOST_FI_OK** 0x00
- #define **HOST_FI_ERROR** 0x01
- #define **HOST_FI_WR_CHECKSUM** 0x02
- #define **HOST_FI_WR_ACT_CODE** 0x03
- #define **HOST_FI_FILE** 0x00
- #define **HOST_FI_DIALOG** 0x01
- #define **HOST_FI_ACTCODE** 0x02
- #define **HOST_FI_NOACTCODE** 0x03
- #define **GUI_FI_LENGTH** 0x01
- #define **GUI_FI_ADDRESS** 0x02
- #define **GUI_FI_CHECKSUM** 0x03
- #define **GUI_FI_DATA** 0x04
- #define **GUI_FI_ACT_CODE** 0x05
- #define **GUI_FI_NBLOCK** 0X06
- #define **GUI_FI_OK** 0x07
- #define **GUI_FI_ERROR** 0x08
- #define **GUI_FI_END** 0x09

4.5.2 Functions

- uint8_t **Gui_LoadFunctionImageCbus_CMX7x3x** (LPC_SSP_T *CBusPort, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC (using dialog) in a CMX7x3x device.
- uint8_t **Gui_LoadFunctionImageCbus_CMX704x** (LPC_SSP_T *CBusPort, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC (using dialog) in a CMX704x device.
- uint8_t **Gui_LoadFunctionImageCbus_CMX714x** (LPC_SSP_T *CBusPort, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC (using dialog) in a CMX714x device.

- uint8_t **Gui_LoadFunctionImageCbus_CMX724x** (LPC_SSP_T *CBusPort, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC (using dialog) in a CMX724x device.
- uint8_t **Gui_LoadFunctionImageCbus_CMX734x** (LPC_SSP_T *CBusPort, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC (using dialog) in a CMX734x device.
- uint8_t **Gui_LoadFunctionImageCbus_CMX7x6x** (LPC_SSP_T *CBusPort, uint16_t *productId, uint16_t *fiVersion)
  Load a FI directly from the PC (using dialog) to a CMX7x6x device.
- uint8_t **Gui_LoadFunctionImageCbus_DirectFile_CMX7x3x** (LPC_SSP_T *CBusPort, const char *filename, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC in a CMX7x3x device without dialog.
- uint8_t **Gui_LoadFunctionImageCbus_DirectFile_CMX704x** (LPC_SSP_T *CBusPort, const char *filename, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC in a CMX704x device.
- uint8_t **Gui_LoadFunctionImageCbus_DirectFile_CMX714x** (LPC_SSP_T *CBusPort, const char *filename, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC in a CMX714x device.
- uint8_t **Gui_LoadFunctionImageCbus_DirectFile_CMX724x** (LPC_SSP_T *CBusPort, const char *filename, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC in a CMX724x device.
- uint8_t **Gui_LoadFunctionImageCbus_DirectFile_CMX734x** (LPC_SSP_T *CBusPort, const char *filename, uint32_t dwActivationCode, uint16_t *productId, uint16_t *fiVersion)
  Load a FI from the PC in a CMX734x device.
- uint8_t **Gui_LoadFunctionImageCbus_DirectFile_CMX7x6x** (LPC_SSP_T *CBusPort, const char *filename, uint16_t *productId, uint16_t *fiVersion)
  Load a FI directly from the PC to a CMX7x6x device.

4.5.3 Macro Definition Documentation

#define CML_RAM_TIME 300000

#define CML_TARGET_MEM 0xABCD

#define CML_TARGET_RAM 0x0001

#define FI_ERROR 1

General error.

#define FI_ERROR2 4

Error, might be wrong FI file path.

#define FI_OK 0

OK.

#define FI_WR_ACT_CODE 3

Wrong activation code.

#define FI_WR_CHECKSUM 2

Wrong checksum.

#define GUI_FI_ACT_CODE 0x05

#define GUI_FI_ADDRESS 0x02

#define GUI_FI_CHECKSUM 0x03

#define GUI_FI_DATA 0x04

#define GUI_FI_END 0x09

#define GUI_FI_ERROR 0x08

#define GUI_FI_LENGTH 0x01

#define GUI_FI_NBLOCK 0X06

#define GUI_FI_OK 0x07

#define HOST_FI_ACTCODE 0x02

#define HOST_FI_DIALOG 0x01

#define HOST_FI_ERROR 0x01

#define HOST_FI_FILE 0x00

#define HOST_FI_FIVERSION 0x07

#define HOST_FI_LOAD_CMX704X_CMX714X 0x02

#define HOST_FI_LOAD_CMX724X_CMX734X 0x03

#define HOST_FI_LOAD_CMX7X3X 0x01

#define HOST_FI_LOAD_CMX7X6X 0x04

#define HOST_FI_NOACTCODE 0x03

#define HOST_FI_OK 0x00

#define HOST_FI_PRODUCTID 0x06

#define HOST_FI_SENDBLOCK 0x08

#define HOST_FI_STATUS 0x05

#define HOST_FI_WR_ACT_CODE 0x03

#define HOST_FI_WR_CHECKSUM 0x02

uint8_t Gui_LoadFunctionImageCbus_CMX704x (LPC_SSP_T * *CBusPort*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC (using dialog) in a CMX704x device.

Parameters:

CBusPort	- CBUS1, CBUS2
dwActivationCode	- Activation code. Example 0xABCD1234
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC. Currently uses a dialog box to get the FI from the PC, Later versions will have direct download of FI from the PC.

Old legacy way for loading one data at the time

uint8_t Gui_LoadFunctionImageCbus_CMX714x (LPC_SSP_T * *CBusPort*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC (using dialog) in a CMX714x device.

Parameters:

CBusPort	- CBUS1, CBUS2
dwActivationCode	- Activation code
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC. Currently uses a dialog box to get the FI from the PC, Later versions will have direct download of FI from the PC.

Old legacy way for loading one data at the time

uint8_t Gui_LoadFunctionImageCbus_CMX724x (LPC_SSP_T * *CBusPort*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC (using dialog) in a CMX724x device.

Parameters:

CBusPort	- CBUS1, CBUS2
dwActivationCode	- Activation code
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC. Currently uses a dialog box to get the FI from the PC, Later versions will have direct download of FI from the PC.

Uses streaming mode.

uint8_t Gui_LoadFunctionImageCbus_CMX734x (LPC_SSP_T * *CBusPort*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC (using dialog) in a CMX734x device.

Parameters:

CBusPort	- CBUS1, CBUS2
dwActivationCode	- Activation code

productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC. Currently uses a dialog box to get the FI from the PC, Later versions will have direct download of FI from the PC.

Uses streaming mode.

uint8_t Gui_LoadFunctionImageCbus_CMX7x3x (LPC_SSP_T * *CBusPort*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC (using dialog) in a CMX7x3x device.

Parameters:

CBusPort	- CBUS1, CBUS2
dwActivationCode	- Activation code
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC. Currently uses a dialog box to get the FI from the PC, Later versions will have direct download of FI from the PC.

Uses a legacy method that loads one data at the time

uint8_t Gui_LoadFunctionImageCbus_CMX7x6x (LPC_SSP_T * *CBusPort*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI directly from the PC (using dialog) to a CMX7x6x device.

Parameters:

CBusPort	- CBUS1, CBUS2
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC. Currently uses a dialog box to get the FI from the PC, Later versions will have direct download of FI from the PC.

Uses streaming mode.

uint8_t Gui_LoadFunctionImageCbus_DirectFile_CMX704x (LPC_SSP_T * *CBusPort*, const char * *filename*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC in a CMX704x device.

Parameters:

CBusPort	- CBUS1, CBUS2
filename	- Path with the FI filename. Must be the full path in your PC or relative path to "MyDocuments\CMLMicro" If "MyDocuments\CMLMicro" does not exist then it is relative to "MyDocuments"
dwActivationCode	- Activation cod. Example 0xABCD1234
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC.

uint8_t Gui_LoadFunctionImageCbus_DirectFile_CMX714x (LPC_SSP_T * *CBusPort*, const char * *filename*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC in a CMX714x device.

Parameters:

CBusPort	- CBUS1, CBUS2
filename	- Path with the FI filename. Must be the full path in your PC or relative path to "MyDocuments\CMLMicro" If "MyDocuments\CMLMicro" does not exist then it is relative to "MyDocuments"
dwActivationCode	- Activation code
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC.

uint8_t Gui_LoadFunctionImageCbus_DirectFile_CMX724x (LPC_SSP_T * *CBusPort*, const char * *filename*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC in a CMX724x device.

Parameters:

CBusPort	- CBUS1, CBUS2
filename	- Path with the FI filename. Must be the full path in your PC or relative path to "MyDocuments\CMLMicro" If "MyDocuments\CMLMicro" does not exist then it is relative to "MyDocuments"
dwActivationCode	- Activation code
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC.

uint8_t Gui_LoadFunctionImageCbus_DirectFile_CMX734x (LPC_SSP_T * *CBusPort*, const char * *filename*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC in a CMX734x device.

Parameters:

CBusPort	- CBUS1, CBUS2
filename	- Path with the FI filename. Must be the full path in your PC or relative path to "MyDocuments\CMLMicro" If "MyDocuments\CMLMicro" does not exist then it is relative to "MyDocuments"
dwActivationCode	- Activation code
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC.

uint8_t Gui_LoadFunctionImageCbus_DirectFile_CMX7x3x (LPC_SSP_T * *CBusPort*, const char * *filename*, uint32_t *dwActivationCode*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI from the PC in a CMX7x3x device without dialog.

Parameters:

CBusPort	- CBUS1, CBUS2
filename	- Path with the FI filename. Must be the full path in your PC or relative path to "MyDocuments\CMLMicro" If "MyDocuments\CMLMicro" does not exist then it is relative to "MyDocuments"
dwActivationCode	- Activation code
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC.

uint8_t Gui_LoadFunctionImageCbus_DirectFile_CMX7x6x (LPC_SSP_T * *CBusPort*, const char * *filename*, uint16_t * *productId*, uint16_t * *fiVersion*)

Load a FI directly from the PC to a CMX7x6x device.

Parameters:

CBusPort	- CBUS1, CBUS2
filename	- Path with the FI filename. Must be the full path in your PC or relative path to "MyDocuments\CMLMicro" If "MyDocuments\CMLMicro" does not exist then it is relative to "MyDocuments"
productId	- Pointer to return the product ID value
fiVersion	- Pointer to return the FI Version value

Returns:

result FI_OK, FI_ERROR, FI_WR_CHECKSUM

Note:

Uses the EC0003 GUI to run a program for downloading a FI from the PC.

4.6 inc/mon_file.h File Reference

4.6.1 Macros

- #define **HOST_FILE_OPEN** 0x01
- #define **HOST_FILE_WRITE** 0x02
- #define **HOST_FILE_READ** 0x03
- #define **HOST_FILE_CLOSE** 0x04
- #define **HOST_FILE_OPENWAV** 0x05
- #define **HOST_FILE_WRITEWAV** 0x06
- #define **HOST_FILE_READWAV** 0x07
- #define **HOST_FILE_CLOSEWAV** 0x08
- #define **GUI_FILE_OPEN** 0x01
- #define **GUI_FILE_WRITE** 0x02
- #define **GUI_FILE_READ** 0x03
- #define **GUI_FILE_CLOSE** 0x04
- #define **GUI_FILE_OPENWAV** 0x05
- #define **GUI_FILE_WRITEWAV** 0x06
- #define **GUI_FILE_READWAV** 0x07
- #define **GUI_FILE_CLOSEWAV** 0x08
- #define **FILE_TYPE_BINARY** 1
- #define **FILE_TYPE_TEXT** 2
- #define **FILE_BYTE** 3
- #define **FILE_WORD** 4
- #define **FILE_DWORD** 5
- #define **FILE_TEXT** 6
- #define **FILE_FORMAT** 7
- #define **FILE_WRITE** 10
- #define **FILE_READ** 11
- #define **FILE_APPEND** 12
- #define **FILE_DIALOGENABLE** 13
- #define **FILE_DIALOGDISABLE** 14
- #define **FILE_WAV** 15
- #define **GUI_FILE_RESULT_ERROR** 0x80
- #define **GUI_FILE_RESULT_OK** 0x00
- #define **GUI_FILE_OVERSIZE_ERROR** 0x81
- #define **FILEDIALOG_ENABLE** 1
- #define **FILEDIALOG_DISABLE** 0
- #define **FILEWAV_MONO** (0 << 4)
- #define **FILEWAV_STEREO** (1 << 4)
- #define **FILEWAV_BITSPERSAMPLE_8B** (0 << 5)
- #define **FILEWAV_BITSPERSAMPLE_16B** (1 << 5)
- #define **FILEWAV_STEREO_MSK** (1 << 4)
- #define **FILEWAV_BITSPERSAMPLER_MSK** (1 << 5)
- #define **FILE_TYPE_BYTE** 1
- #define **FILE_TYPE_WORD** 2
- #define **FILE_TYPE_DWORD** 3
- #define **FILE_TYPE_STRING** 4
- #define **MAX_PACKETSIZE** 0x1000

4.6.2 Functions

- uint8_t **Gui_FileTxtOpen** (uint8_t *file_id, const char *filename, uint8_t fileOperation, const char *format)
  Open a TXT file and use the formatter to get/set the data.
- uint8_t **Gui_FileTxtDialogOpen** (uint8_t *file_id, const char *filename, uint8_t fileOperation, const char *format)
  Open a TXT file and use the formatter to get/set the data. Uses a dialog to get/set the file.

- uint8_t **Gui_FileBinOpen** (uint8_t *file_id, const char *filename, uint8_t fileOperation)
  Open a binary file and uses the formatter to get/set the data.
- uint8_t **Gui_FileBinDialogOpen** (uint8_t *file_id, const char *filename, uint8_t fileOperation)
  Open a binary file and uses the formatter to get/set the data. Uses a dialog to select the file.
- uint8_t **Gui_FileTxtWrite** (uint8_t file_id, uint8_t *data, uint32_t length, uint32_t *nDataWritten, uint8_t type)
  Write data to a txt file.
- uint8_t **Gui_FileTxtRead** (uint8_t file_id, uint8_t *data, uint32_t length, uint32_t *nDataRead, uint8_t type)
  Read a Txt file.
- uint8_t **Gui_FileWriteByte** (uint8_t file_id, uint8_t *data, uint32_t length, uint32_t *nDataWritten)
  Write bytes of data into a binary file.
- uint8_t **Gui_FileReadByte** (uint8_t file_id, uint8_t *data, uint32_t length, uint32_t *nDataRead)
  Read bytes from a file.
- uint8_t **Gui_FileWriteWord** (uint8_t file_id, uint16_t *data, uint32_t length, uint32_t *nDataWritten)
  Write words into a binary file.
- uint8_t **Gui_FileReadWord** (uint8_t file_id, uint16_t *data, uint32_t length, uint32_t *nDataRead)
  Read Words from a file.
- uint8_t **Gui_FileWriteDWord** (uint8_t file_id, uint32_t *data, uint32_t length, uint32_t *nDataWritten)
  Write Dwords into a binary file.
- uint8_t **Gui_FileReadDWord** (uint8_t file_id, uint32_t *data, uint32_t length, uint32_t *nDataRead)
  Read Dwords from a file.
- uint8_t **Gui_FileClose** (uint8_t file_id)
  Close file.
- uint8_t **Gui_FileWavOpen** (uint8_t *file_id, const char *filename, uint8_t fileOperation, uint8_t isStereo, uint8_t bitsPerSample, uint32_t sampleRate)
  Open a wav file.
- uint8_t **Gui_FileWavDialogOpen** (uint8_t *file_id, const char *filename, uint8_t fileOperation, uint8_t isStereo, uint8_t bitsPerSample, uint32_t sampleRate)
  Open a wav file selected via open dialog.
- uint8_t **Gui_FileWavWrite** (uint8_t file_id, uint8_t *data, uint32_t length, uint32_t *nDataWritten, uint8_t bitsPerSample)
  Write samples into a wav file.
- uint8_t **Gui_FileWavWriteByteSamples** (uint8_t file_id, uint8_t *data, uint32_t length, uint32_t *nDataWritten)
  Write byte samples into a wav file.
- uint8_t **Gui_FileWavWriteWordSamples** (uint8_t file_id, uint16_t *data, uint32_t length, uint32_t *nDataWritten)
  Write word samples into a wav file.
- uint8_t **Gui_FileWavRead** (uint8_t file_id, uint8_t *data, uint32_t length, uint32_t *nDataRead, uint8_t bitsPerSample)
  Read samples from a wav file.
- uint8_t **Gui_FileWavReadByteSamples** (uint8_t file_id, uint8_t *data, uint32_t length, uint32_t *nDataRead)
  Read samples from a wav file.
- uint8_t **Gui_FileWavReadWordSamples** (uint8_t file_id, uint16_t *data, uint32_t length, uint32_t *nDataRead)
  Read samples from a wav file.
- uint8_t **Gui_FileWavClose** (uint8_t file_id)
  Close the fil_id wav file previously opened.

4.6.3 Macro Definition Documentation

#define FILE_APPEND 12

#define FILE_BYTE 3

#define FILE_DIALOGDISABLE 14

#define FILE_DIALOGENABLE 13

#define FILE_DWORD 5

#define FILE_FORMAT 7

#define FILE_READ 11

#define FILE_TEXT 6

#define FILE_TYPE_BINARY 1

#define FILE_TYPE_BYTE 1

#define FILE_TYPE_DWORD 3

#define FILE_TYPE_STRING 4

#define FILE_TYPE_TEXT 2

#define FILE_TYPE_WORD 2

#define FILE_WAV 15

#define FILE_WORD 4

#define FILE_WRITE 10

#define FILEDIALOG_DISABLE 0

#define FILEDIALOG_ENABLE 1

```
#define FILEWAV_BITSPERSAMPLE_16B  (1 << 5)

#define FILEWAV_BITSPERSAMPLE_8B  (0 << 5)

#define FILEWAV_BITSPERSAMPLER_MSK  (1 << 5)

#define FILEWAV_MONO  (0 << 4)

#define FILEWAV_STEREO  (1 << 4)

#define FILEWAV_STEREO_MSK  (1 << 4)

#define GUI_FILE_CLOSE  0x04

#define GUI_FILE_CLOSEWAV  0x08

#define GUI_FILE_OPEN  0x01

#define GUI_FILE_OPENWAV  0x05

#define GUI_FILE_OVERSIZE_ERROR  0x81

#define GUI_FILE_READ  0x03

#define GUI_FILE_READWAV  0x07

#define GUI_FILE_RESULT_ERROR  0x80

#define GUI_FILE_RESULT_OK  0x00

#define GUI_FILE_WRITE  0x02

#define GUI_FILE_WRITEWAV  0x06

#define HOST_FILE_CLOSE  0x04

#define HOST_FILE_CLOSEWAV  0x08

#define HOST_FILE_OPEN  0x01
```

#define HOST_FILE_OPENWAV 0x05

#define HOST_FILE_READ 0x03

#define HOST_FILE_READWAV 0x07

#define HOST_FILE_WRITE 0x02

#define HOST_FILE_WRITEWAV 0x06

#define MAX_PACKETSIZE 0x1000

4.6.4 Function Documentation

uint8_t Gui_FileBinDialogOpen (uint8_t * *file_id*, const char * *filename*, uint8_t *fileOperation*)

Open a binary file and uses the formatter to get/set the data. Uses a dialog to select the file.

Parameters:

file_id	- Pointer to return the file ID.
filename	- Name of the file to read/write
fileOperation	- File open method: FILE_WRITE, FILE_READ, FILE_APPEND

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

uint8_t Gui_FileBinOpen (uint8_t * *file_id*, const char * *filename*, uint8_t *fileOperation*)

Open a binary file and uses the formatter to get/set the data.

Parameters:

file_id	- Pointer to return the file ID.
filename	- Name of the file
fileOperation	- File open method: FILE_WRITE, FILE_READ, FILE_APPEND

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

uint8_t Gui_FileClose (uint8_t *file_id*)

Close file.

Parameters:

file_id	- File identifier

Returns:

uint8_t Gui_FileReadByte (uint8_t *file_id*, uint8_t * *data*, uint32_t *length*, uint32_t * *nDataRead*)

Read bytes from a file.

Parameters:

file_id	- File identifier
data	- Pointer to the data store
length	- number of data to read.
nDataRead	- Return the number of data read.

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must first be opened for reading Generic function used by target specific functions LIMIT SIZE OF CHUNKS TO 4096 bytes

uint8_t Gui_FileReadDWord (uint8_t *file_id*, uint32_t * *data*, uint32_t *length*, uint32_t * *nDataRead*)

Read Dwords from a file.

Parameters:

file_id	- File identifier
data	- Pointer to the data store
length	- number of data.
nDataRead	- Return the number of data read.

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must be open for reading before executing LIMIT SIZE OF CHUNKS TO 4096 bytes

uint8_t Gui_FileReadWord (uint8_t *file_id*, uint16_t * *data*, uint32_t *length*, uint32_t * *nDataRead*)

Read Words from a file.

Parameters:

file_id	- File identifier
data	- Pointer to the data store
length	- number of data.
nDataRead	- Return the number of data read.

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must be open for reading before executing Generic function used by target specific functions
LIMIT SIZE OF CHUNKS TO 4096 bytes

uint8_t Gui_FileTxtDialogOpen (uint8_t * *file_id*, const char * *filename*, uint8_t *fileOperation*, const char * *format*)

Open a TXT file and use the formatter to get/set the data. Uses a dialog to get/set the file.

Parameters:

file_id	- Pointer to return the file ID.
filename	- Name of the file
fileOperation	- File open method: FILE_WRITE, FILE_READ, FILE_APPEND
format	- String with the required data formatter

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

Currently, the formatter accepts only one parameter per line

uint8_t Gui_FileTxtOpen (uint8_t * *file_id*, const char * *filename*, uint8_t *fileOperation*, const char * *format*)

Open a TXT file and use the formatter to get/set the data.

Parameters:

file_id	- Pointer to return the file ID.

filename	- Name of the file
fileOperation	- File open method: FILE_WRITE, FILE_READ, FILE_APPEND
format	- String with the required data formatter

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

Currently, the formatter accepts only one parameter per line. Format example "%d ,"

uint8_t Gui_FileTxtRead (uint8_t *file_id*, uint8_t * *data*, uint32_t *length*, uint32_t * *nDataRead*, uint8_t *type*)

Read a Txt file.

Parameters:

file_id	- File identifier
data	- Pointer to data store
length	- number of data to read
nDataRead	- Return the number of data read.
type	- FILE_TYPE_BYTE, FILE_TYPE_WORD, FILE_TYPE_DWORD

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must first be opened for reading

uint8_t Gui_FileTxtWrite (uint8_t *file_id*, uint8_t * *data*, uint32_t *length*, uint32_t * *nDataWritten*, uint8_t *type*)

Write data to a txt file.

Parameters:

file_id	- Pointer to return the file ID.
data	- Pointer to the data source(data types: uint8_t, uint16_t, uint32_t)
length	- number of data to write
nDataWritten	- Return the number of data written.
type	- Data size FILE_TYPE_BYTE, FILE_TYPE_WORD, FILE_TYPE_DWORD, FILE_TYPE_STRING

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

The file must first be opened for writing or appending

uint8_t Gui_FileWavClose (uint8_t *file_id*)

Close the fil_id wav file previously opened.

Parameters:

file_id	- File identifier.

Returns:

TRUE

uint8_t Gui_FileWavDialogOpen (uint8_t * *file_id*, const char * *filename*, uint8_t *fileOperation*, uint8_t *isStereo*, uint8_t *bitsPerSample*, uint32_t *sampleRate*)

Open a wav file selected via open dialog.

Parameters:

file_id	- Pointer to return the file Id
filename	- Name of the file
fileOperation	- File open method: FILE_WRITE, FILE_READ, FILE_APPEND
isStereo	- Is stereo or mono. Values - FILEWAV_STEREO, FILEWAV_MONO
bitsPerSample	- Bumber of bits per sample. Values - FILEWAV_BITSPERSAMPLE_8B, FILEWAV_BITSPERSAMPLE_16B
sampleRate	- Sample rate

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

Generic function used by specific targeted functions Gui_FileWavOpen - Wav file
Gui_FileWavDialogOpen - Wav file with using an open dialog

uint8_t Gui_FileWavOpen (uint8_t * *file_id*, const char * *filename*, uint8_t *fileOperation*, uint8_t *isStereo*, uint8_t *bitsPerSample*, uint32_t *sampleRate*)

Open a wav file.

Parameters:

file_id	- Pointer to return the file Id
filename	- Name of the file
fileOperation	- File open method: FILE_WRITE, FILE_READ, FILE_APPEND
isStereo	- Is stereo or mono. Values - FILEWAV_STEREO, FILEWAV_MONO
bitsPerSample	- Bumber of bits per sample. Values - FILEWAV_BITSPERSAMPLE_8B, FILEWAV_BITSPERSAMPLE_16B
sampleRate	- Sample rate

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

Generic function used by specific targeted functions Gui_FileWavOpen - Wav file Gui_FileWavDialogOpen - Wav file with using an open dialog

uint8_t Gui_FileWavRead (uint8_t *file_id*, uint8_t * *data*, uint32_t *length*, uint32_t * *nDataRead*, uint8_t *bitsPerSample*)

Read samples from a wav file.

Parameters:

file_id	- File identifier
data	- Pointer with the data
length	- number of data.
nDataRead	- number of data read
bitsPerSample	- Bits per sample FILEWAV_BITSPERSAMPLE_8B, FILEWAV_BITSPERSAMPLE_16B

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must be open for reading before executing Generic function used by target specific functions

uint8_t Gui_FileWavReadByteSamples (uint8_t *file_id*, uint8_t * *data*, uint32_t *length*, uint32_t * *nDataRead*)

Read samples from a wav file.

Parameters:

file_id	- File identifier
data	- Pointer with the data
length	- number of data.
nDataRead	- Number of data read

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must be open for reading before executing Generic function used by target specific functions

uint8_t Gui_FileWavReadWordSamples (uint8_t *file_id*, uint16_t * *data*, uint32_t *length*, uint32_t * *nDataRead*)

Read samples from a wav file.

Parameters:

file_id	- File identifier
data	- Pointer with the data
length	- number of data.
nDataRead	- Number of words read

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must be open for reading before executing Generic function used by target specific functions

uint8_t Gui_FileWavWrite (uint8_t *file_id*, uint8_t * *data*, uint32_t *length*, uint32_t * *nDataWritten*, uint8_t *bitsPerSample*)

Write samples into a wav file.

Parameters:

file_id	- File identifier
data	- Pointer with the data, first data address to store (it can be uint8_t, uint16_t, uint32_t)
length	- number of data.
nDataWritten	- Return the number of data written.

bitsPerSample	- Number the bits per sample. Values FILEWAV_BITSPERSAMPLE_8B, FILEWAV_BITSPERSAMPLE_16B

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

Write binary data. Generic function used by specific target functions Gui_FileWriteByte Gui_FileWriteWord Gui_FileWriteDWord

uint8_t Gui_FileWavWriteByteSamples (uint8_t *file_id*, uint8_t * *data*, uint32_t *length*, uint32_t * *nDataWritten*)

Write byte samples into a wav file.

Parameters:

file_id	- File identifier
data	- Pointer with the data, first data address to store (it can be uint8_t, uint16_t, uint32_t)
length	- number of data.
nDataWritten	- Return the number of data written.

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

Write binary data. Generic function used by specific target functions Gui_FileWriteByte Gui_FileWriteWord Gui_FileWriteDWord

uint8_t Gui_FileWavWriteWordSamples (uint8_t *file_id*, uint16_t * *data*, uint32_t *length*, uint32_t * *nDataWritten*)

Write word samples into a wav file.

Parameters:

file_id	- File identifier
data	- Pointer with the data, first data address to store (it can be uint8_t, uint16_t, uint32_t)
length	- number of data.
nDataWritten	- Return the number of data written.

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

Write binary data. Generic function used by specific target functions Gui_FileWriteByte Gui_FileWriteWord Gui_FileWriteDWord

uint8_t Gui_FileWriteByte (uint8_t *file_id*, uint8_t * *data*, uint32_t *length*, uint32_t * *nDataWritten*)

Write bytes of data into a binary file.

Parameters:

file_id	- File identifier
data	- Pointer to the data source
length	- number of data to write.
nDataWritten	- Return the number of data written.

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must first be opened for writing

uint8_t Gui_FileWriteDWord (uint8_t *file_id*, uint32_t * *data*, uint32_t *length*, uint32_t * *nDataWritten*)

Write Dwords into a binary file.

Parameters:

file_id	- File identifier
data	- Pointer to the data source
length	- number of data.
nDataWritten	- Return the number of data written.

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must be open for writing before executing. LIMIT SIZE OF CHUNKS TO 4096 bytes

uint8_t Gui_FileWriteWord (uint8_t *file_id*, uint16_t * *data*, uint32_t *length*, uint32_t * *nDataWritten*)

Write words into a binary file.

Parameters:

file_id	- File identifier
data	- Pointer to the data source
length	- number of data to write.
nDataWritten	- Return the number of data written.

Returns:

result - GUI_FILE_RESULT_OK, GUI_FILE_RESULT_ERROR

Note:

File must be open for writing before executing. LIMIT SIZE OF CHUNKS TO 4096 bytes

4.7 inc/mon_general.h File Reference

```
#include "lpc_types.h"
#include <stdio.h>
#include <stdarg.h>
```

4.7.1 Functions

- int **mprintf** (const char *format,...)
  Print to the EC0003 console.
- int **mscanf** (const char *format,...)
  Read input from EC0003 console.
- void **EnableNonBlockingScanf** ()
  Enables NonBlockingScanf.
- void **DisableNonBlockingScanf** ()
  Disables NonBlockingScanf.
- void **Gui_ClearDisplay** ()
  Clear the EC0003 console panel.
- void **Gui_ErrorReport** (const char *error)
  Report an error during execution. Pop up a window in the GUI to display the message.
- void **Gui_ErrorReportBlocking** (const char *error)
  Report an error during execution and Block the execution.
- void **Gui_EndOfProgram** ()
  Report the end of the program.

4.7.2 Function Documentation

void DisableNonBlockingScanf ()

Disables NonBlockingScanf.

Returns:
none

Note:
Set the normal behaviour of the scanf operation

void EnableNonBlockingScanf ()

Enables NonBlockingScanf.

Returns:
none

Note:
The non-blocking scanf. Read the input stream if there is some data to read otherwise ignore it. Placing a non-blocking scanf in a loop allows the loop runs without stopping but reading and processing input data on each pass.

void Gui_ClearDisplay ()

Clear the EC0003 console panel.

Returns:

none

void Gui_EndOfProgram ()

Report the end of the program.

Returns:

none

void Gui_ErrorReport (const char * *error*)

Report an error during execution. Pop up a window in the GUI to display the message.

Parameters:

error	- error string

Returns:

none

void Gui_ErrorReportBlocking (const char * *error*)

Report an error during execution and Block the execution.

Parameters:

error	- error string

Returns:

none

int mprintf (const char * *format*, ...)

Print to the EC0003 console.

Parameters:

format	n formatted parameters

Returns:

<0 error

Note:

Works in the same way as the standard printf with the same modifiers and descriptors

```
mprintf("Number %d", 8);
mprintf("Number %d %s", 8, "bits");
```

int mscanf (const char * *format*, ...)

Read input from EC0003 console.

Parameters:

format	n variables

Returns:

<0 error, 0 nothing to process(in non-blocking mode), number data processed

Note:

Equivalent to standard scanf. Uses the same modifiers and descriptors Two modes of use:

- The traditional one. When the function is called, a message is sent to the GUI and a response is waited for.
- Non-Blocking mode. Call checks are made to an internal ring buffer looking for values that match the required format. To enable this mode, execute the function EnableNonBlockingScanf before calling. To disable this mode, execute the function DisableNonBlockingScanf. When the mode is enabled the GUI EC0003 console allows to insert characters that are buffered in the PE0003 and processed whenever the function is called. So if there is a non-blocking scanf in a loop the loop runs without stopping and can read inputs on the fly.

```
char c;
    mscanf("Read key from GUI %c", &c);
```

4.8 inc/monitor.h File Reference

```
#include "lpc_types.h"
```

4.8.1 Macros

- #define **CML_GUI_ENABLE**

4.8.2 Functions

- void **Cml_GuiSystemInit** ()
  Initialize communications with the GUI and establish the link.
- void **Cml_ResetGuiMessage** ()
  Generates a reset of the PE0003 by software.
- void **Cml_MessageSend** (uint8_t msg, uint32_t length, uint8_t *data)
  Basic function to send messages to the EC0003 GUI.
- uint8_t **Cml_MessageReceive** (uint8_t msg, uint32_t *length, uint8_t *data)
  Basic function to receive messages from the EC0003 GUI.
- uint8_t **Cml_MessageReceiveAsync** (uint8_t msg, uint32_t *length, uint8_t *data, uint8_t blocking)
  Basic function to receive asynchronous messages from the EC0003 GUI.

4.8.3 Macro Definition Documentation

#define CML_GUI_ENABLE

4.8.4 Function Documentation

void Cml_GuiSystemInit ()

Initialize communications with the GUI and establish the link.

Returns:
None

Note:
This function must be executed to work with the EC0003 GUI system The init of the board must be done before calling this function

- Requires the USB FTDI connection

State machine three states Blocking operation there is a while loop that holds the control until the connection with GUI is established

Warning:
THIS FUNCTION MUST BE EXECUTED TO MAKE USE OF THE GUI LIBRARIES

uint8_t Cml_MessageReceive (uint8_t *msg*, uint32_t * *length*, uint8_t * *data*)

Basic function to receive messages from the EC0003 GUI.

Parameters:

msg	- Message. Check **messages.h** for messages

length	- Dataframe length
data	- Received data in a byte array

Returns:

message result MSG_ERROR, MSG_OK

uint8_t Cml_MessageReceiveAsync (uint8_t *msg*, uint32_t * *length*, uint8_t * *data*, uint8_t *blocking*)

Basic function to receive asynchronous messages from the EC0003 GUI.

Parameters:

msg	- Message.
length	- Dataframe length
data	- Received data in a byte array
blocking	- true for blocking, false unblocking

Returns:

message result MSG_ERROR, MSG_OK

Note:

Basically works like the normal Message_receive but it checks the flagCommand to verify if the command flag is set for that specific command and reads the data from the async ring buffer instead of the ring buffer.

Used to run asynchronous operations. Basically operations initiated from the EC0003 instead the PE0003.

void Cml_MessageSend (uint8_t *msg*, uint32_t *length*, uint8_t * *data*)

Basic function to send messages to the EC0003 GUI.

Parameters:

msg	- Message. Type of the message, check **messages.h** for messages
length	- Dataframe length
data	- Data to send into a byte array

Returns:

none

void Cml_ResetGuiMessage ()

Generates a reset of the PE0003 by software.

Returns:

None

CML Microcircuits (UK) Ltd *COMMUNICATION SEMICONDUCTORS*	**CML Microcircuits (USA) Inc.** *COMMUNICATION SEMICONDUCTORS*	**CML Microcircuits (Singapore) Pte Ltd** *COMMUNICATION SEMICONDUCTORS*
Tel: +44 (0)1621 875500 **Fax:** +44 (0)1621 875600 **Sales:** sales@cmlmicro.com **Tech Support:** techsupport@cmlmicro.com	**Tel:** +1 336 744 5050 800 638 5577 **Fax:** +1 336 744 5054 **Sales:** us.sales@cmlmicro.com **Tech Support:** us.techsupport@cmlmicro.com	**Tel:** +65 62 888129 **Fax:** +65 62 888230 **Sales:** sg.sales@cmlmicro.com **Tech Support:** sg.techsupport@cmlmicro.com
- www.cmlmicro.com -		