AN11342

How to Scale Down the NXP Reader Library

Rev.1.0 — 11 March 2013
259610

Application note
COMPANY PUBLIC

Document information

Info	Content
Keywords	NXP Reader Library, LPCXpresso, LPC1227, CLRC663, MIFARE Classic
Abstract	This application note is related to the procedure of the NXP Reader Library reduction. Reduction is the scale down process where application layer is close to real used hardware.

Revision history

Rev	Date	Description
1.0	20130311	First release

Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

1. NXP Reader Library scale down procedure

1.1 Introduction

This document describes the procedure of reducing the size of the NXP Reader Library source code.

It consists of 3 parts that describe how to reduce the code of application and the NXP Reader Library.

The first part is about typical user application setup.

The second part focuses on the replacement of calling functions with reference to input parameters and the input data structures of the replaced functions. It also tells you how to reach the minimum modules activation in order to reduce the final software package amount.

The third part is an extended library reduction. It deals with source code reduction. This means removing unused parts of the called functions to only keep the necessary required functionality. The part also contains descriptions on how to remove library cross references.

All in all the **reduction** could save up to **33% of the FLASH** memory occupied by the library.

2. Application setup

2.1 Components

The procedure of scaling down focuses on MIFARE Classic card operations and the reader IC CLRC663. The procedure is as follows:

Software setup:

LPCXpresso v4.2.2_275	Installed development environment [2]
RC663-LPC1227-Classic	Project example [3]

Hardware setup:

LPC1227 LPCXpresso Board	microcontroller development kit [4], [5]
CLEV663B blueboard	contactless card reader board [6]

The general URL to the product is to be found here [1]

2.2 Required project setup for the code reduction

The common software example RC663-LPC1227-Classic [3] is based on the microcontroller LPC1227 and reader IC CLRC663. It supports management (authentication, read, write operations, etc. …) of MIFARE Classic card compliant with ISO/IEC 14443 Type A and detection of a wide range of the other MIFARE cards (MIFARE Ultralight, MIFARE Plus, DESFire EV1, ICODE).The communication interface between the reader IC and the microcontroller can either be set to SPI or I^2C protocol.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

The reduced example in this document only uses the SPI communication interface. It supports Low Power Card Detection (LPCD). The card management is reduced to only the MIFARE Classic, compliant with ISO/IEC 14443 Type A standard.

2.3 Project build setup

The build setup and functionality is set in the *…/types/ph_NxpBuild.h* file. In this file one can define which modules to include into the build or exclude from the build.

BAL components have to be defined by:

```
#define NXPBUILD__PHBAL_REG_STUB                    /* Stub BAL definition */
```

HAL components have to be defined by:

```
#define NXPBUILD__PHHAL_HW_RC663                    /* Rc663 HAL definition */
```

PAL ISO 14443-3A components have to be defined by:

```
#define NXPBUILD__PHPAL_I14443P3A_SW   /* Software PAL ISO 14443-3A definition */
```

PAL MIFARE components have to be defined by:

```
#define NXPBUILD__PHPAL_MIFARE_SW                  /**< Software PAL MIFARE */
```

AL MIFARE Classic components have to be defined by:

```
#define NXPBUILD__PHAL_MFC_SW                  /**< Software MIFARE Classic */
```

KeyStore components have to be defined by:

```
#define NXPBUILD__PH_KEYSTORE_RC663                  /**< RC663 KeyStore */
```

This project setup corresponds to the overview by chapter 2.

2.4 General principle of the reduction

In general, when calling hardware dependent functions, the target function is not called directly, but a stub function is called instead. That stub function decides about what hardware dependent function needs to be call.

The principle of the reduction is to use functions directly for the reader IC CLRC663, without the need of the overhead from the stub functions. Because we focus on the MIFARE Classic card we are using only the ISO/IEC 14443 Type A module.

That means the generic interface functions have to be replaced by specific interface function. The whole overview is shown in the Fig 1. The software modules of the NXP Reader Library that are used for the application [3] are highlighted. Any other software modules are not necessary. The hardware platform (LPC1227, CLRC663) is connected through BAL and the direction to the application layer is marked by arrows.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

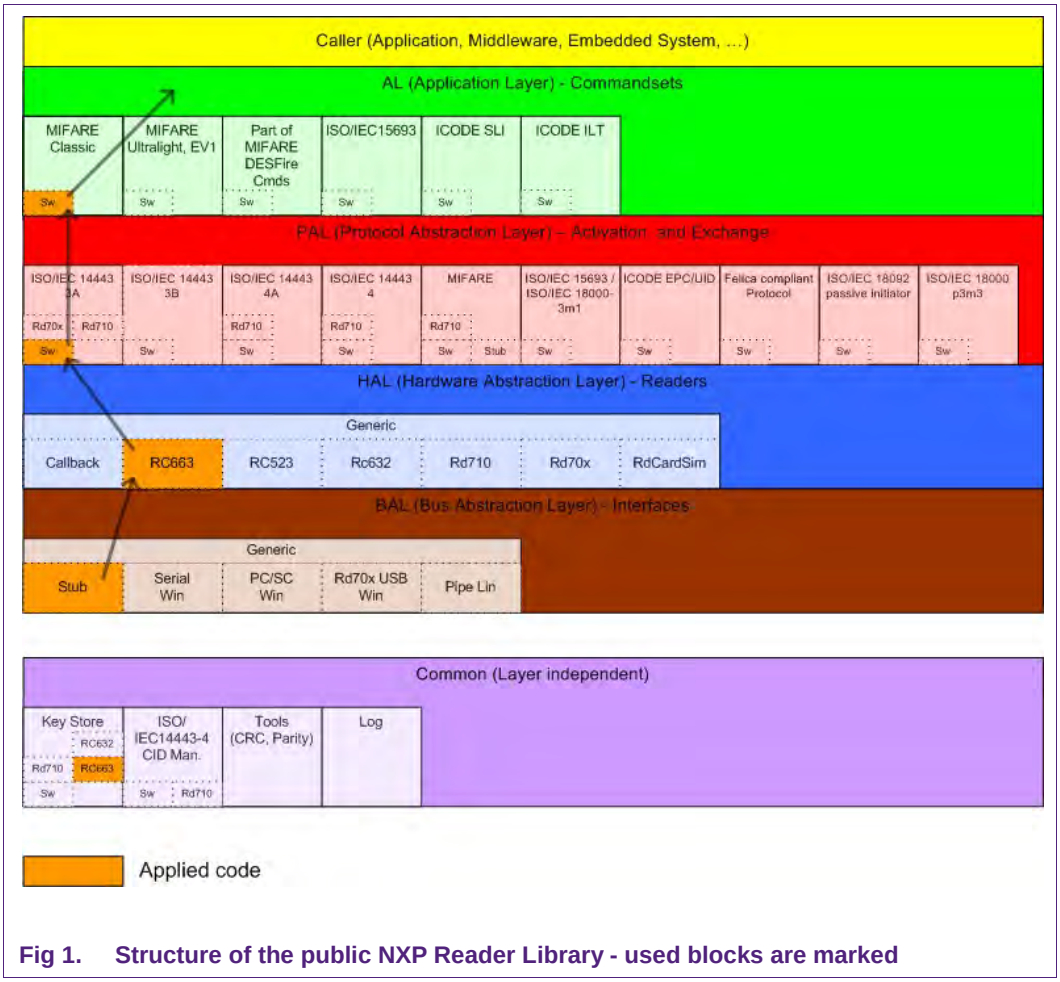

Fig 1. Structure of the public NXP Reader Library - used blocks are marked

2.5 Interrupt handlers

Remove all the unnecessary interrupt handlers, which are not directly used by the user application. The compiler doesn't require interrupt handler definition at all.

For example:

Usually 3 types (SPI, I²C and UART) of communication interfaces between microcontroller and reader IC are defined. It is only possible to select one single communication interface at a time at reader IC (usually SPI). The second interrupt handler has to be removed. The same rule can be applied to the other interrupt handlers.

2.6 Printout

For the final release version of the software it is necessary to remove all printouts. This can be executed as follows:

For example:

```
#if DEBUG
#include <stdio.h>
#endif   //DEBUG
…..
#if DEBUG
        printf("start\n");
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

```
#endif   //DEBUG
```

The second possibility is to use the printout by built in functions or functions from the driver which have integrated conditional compilation.

2.7 Logging

The logging functionality is defined in the *…/types/ph_NxpBuild.h* file. However it may not be defined when working in an embedded environment. This can also consume some substantial flash size.

The usage of the LOG module requires a large RAM.

Main usage of the LOG module is intended for embedded systems based on operating system or other embedded operating system platforms directly.

LOG components have to be undefined as follow:

```
//#define NXPBUILD__PH_LOG
```

3. Library setup

3.1 Generic interface reduction

The NXP Reader Library has a generic interface and specific interfaces for better readability and better usability. For reducing the overhead of the generic interface, the following procedure needs to be followed.

For example:

In the phalMfc component, *phalMfc.h* defines the generic interface and the relevant parameter data structures. If only software implementation is desired, then the file *phalMfc.h* should not have the generic function prototypes (for example *phalMfc_Read()*). It should only have the required parameter data structure definition for SW (*phalMfc_Sw_DataParams_t*) and the other required *#defines*.

The above is only an example for an AL component. The same procedures need to be carried out for PAL, HAL and BALs to scale down the reader library and include the necessary code for the target application.

In the NXP Reader Library, if generic interfaces are removed for scaling down purposes, calls to generic interfaces should be replaced by the relevant specific interface call.

For example:

Function *phhalHw_SetConfig()* should be replaced by *phhalHw_Rc663_SetConfig()*.

3.2 HAL Generic interface reduction - CLRC663 setup

Replace all the functions of the general HAL layer by functions of the applied HAL module.

The functions in the file *phhalHw.c* should be replaced by the functions from the following modules in the RC663 directory:

phhalHw_Rc663.c

phhalHw_Rc663_Int.c

phhalHw_Rc663_Cmd.c

phhalHw_Rc663_Wait.c

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

For example:

These 4 general functions are called very often from the library:

```
phhalHw_SetConfig()
phhalHw_GetConfig()
phhalHw_WriteRegister()
phhalHw_ReadRegister()
```

The functions can be replaced by the following prototypes:

```
phhalHw_Rc663_SetConfig()
phhalHw_Rc663_GetConfig()
phhalHw_Rc663_WriteRegister()
phhalHw_Rc663_ReadRegister()
```

3.3 BAL Generic interface reduction - CLRC663 setup

Replace all the functions of the general BAL layer by functions of the applied STUB module.

The functions in the source file *phbalReg.c* should be replaced by the function of the following modules in the STUB directory:

phbalReg_Stub.c

For example:

Function *phbalReg_SetPort()* should be replaced by *phbalReg_Stub_SetPort().*

3.4 PAL Generic interface reduction - CLRC663 setup

Replace all the functions of the general PAL layer by functions of the applied SW module.

Generally, MIFARE Classic card operations could use the following modules”

phpalI14443p3a.c

phpalI14443p3b.c

phpalI14443p4.c

phpalI14443p4a.c

phpalMifare.c

Functions in these modules should be replaced by functions of modules in/from the SW directory for each integrated protocol.

For example:

Function *phpalI14443p3a_RequestA()* should be replaced by function *phpalI14443p3a_Sw_RequestA()*.

3.5 AL Generic interface reduction - CLRC663 setup

Replace all the functions of the general AL layer by functions of the applied SW module.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

The functions in the file *phalMfc.c* should be replaced by functions from the following modules in/from the SW directory

phalMfc_Sw.c

For example:

Function *phalMfc_Authenticate()* should be replaced by *phalMfc_Sw_Authenticate()*.

NOTE:

Module *phalMfc.c* is special. Functions from this module are called by the AL and call functions from the PAL and these functions in turn call functions from the HAL. These cross connections could be removed and called functions from the PAL could be replaced by HAL functions directly. However, this is an extended reduction and is described in chapter 4.

3.6 Removing macros

In the library each function returns a result operation value. For this operation in most cases the macro *PH_CHECK_SUCCESS_FCT(status,fct)* is used (file …/types/*ph_Status*). Each macro PH_CHECK_SUCCESS_FCT also includes one decision condition that adds another 4 bytes to the code size. For example if the macro is used for 100 calling functions, the final code size could increase up to additional 400 byte.

Replace macro usage by parametric return value based on the following example:

For example:

Before macro removing:

```
phStatus_t status;
……
/* Reset the Rf field */
PH_CHECK_SUCCESS_FCT(status, phhalHw_FieldReset(pHal));
```

After macro removing:

```
phStatus_t status;
……
/* Reset the Rf field */
status = phhalHw_FieldReset(pHal);
```

After final correction:

```
phStatus_t status;
……
/* Reset the Rf field */
status = phhalHw_Rc663_FieldReset(pHal);
```

3.7 Final compilation and FLASH memory saving

The build setup of the project example (2.3) is set in the file *…/intfs/phhalHw.h*.

The conditions have to be defined as follows:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

```
#define SCALE_DOWN_RC663
#ifdef SCALE_DOWN_RC663
//#define SCALE_DOWN_RC663_EXTENDED
#endif
```

Another necessary correction in the changed modules is to add the *phhalHw.h* header file. (NOTE: this correction is already done for this example.)

Reduction of the code by the use of the changed library setup is 13.67%

The code size has been reduced from 0x4492 (17554 bytes) to 0x3b3e (15166 bytes).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

4. Extended library reduction

4.1 Introduction

The extended library reduction is based on removing code that is not directly used for the application. The build setup is set in the file …*/intfs/phhalHw.h*.

The conditions have to be defined as follows:

```
#define SCALE_DOWN_RC663
#ifdef SCALE_DOWN_RC663
#define SCALE_DOWN_RC663_EXTENDED
#endif
```

4.2 Controlled IC reader communication interface

There are 2 functions from the HAL, which directly communicate with the IC reader via microcontroller. They are:

phhalHw_Rc663_ReadRegister()

phhalHw_Rc663_WriteRegister()

A conditional section has been implemented which chooses the communication interface. This section is necessary to reduce the code to one specific interface (file *phhalHw_Rc663.c*). The chosen interface in this example is SPI.

For example:

If the SPI protocol is generally used, the code reduction could look like the following:

```
#ifndef SCALE_DOWN_RC663_EXTENDED
    /* RS232 protocol */
    if (pDataParams->bBalConnectionType == PHHAL_HW_BAL_CONNECTION_RS232)
        {
        /* set RD/NWR bit to indicate read operation */
        bTxBuffer[0] = bAddress | 0x80U;
        wTxLength = 1;
        bNumExpBytes = 1;
        }
    /* SPI protocol */
    else if (pDataParams->bBalConnectionType == PHHAL_HW_BAL_CONNECTION_SPI)
        {
        /* set RD/NWR bit to indicate read operation */
        bTxBuffer[0] = (uint8_t)(bAddress << 1) | 0x01U;
#if 0
        bTxBuffer[1] = 0x00;
        wTxLength = 2;
#endif
        wTxLength = 1;
        bNumExpBytes = 2;
        }
    /* I2C protocol */
    else if (pDataParams->bBalConnectionType == PHHAL_HW_BAL_CONNECTION_I2C)
        {
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

```
            /* nothing to be modified */
            bTxBuffer[0] = bAddress;
            wTxLength = 1;
            bNumExpBytes = 1;
            }
        else
            {
            /* Insert Code for other protocols here */
            wTxLength = 0;
            bNumExpBytes = 0;
            }
    #else // SCALE_DOWN_RC663_EXTENDED
            /* SPI protocol */
            /* set RD/NWR bit to indicate read operation */
            bTxBuffer[0] = (uint8_t)(bAddress << 1) | 0x01U;
            wTxLength = 1;
            bNumExpBytes = 2;
    #endif // SCALE_DOWN_RC663_EXTENDED
```

4.3 Controlled IC reader register setting

There are 4 functions in the HAL (module *phhalHw_Rc663.c*), which are directly needed for the IC reader setting:

phhalHw_Rc663_SetConfig()

phhalHw_Rc663_GetConfig()

phhalHw_Rc663_SetConfig_Int()

phhalHw_Rc663_ApplyProtocolSettings()

Basically, the functions are big switches and the direct IC reader register setting depends on the input parameters for internal case - switch conditions. Each case represents one parameter setting or a group of actions that set one hardware parameter.

It is necessary to remove all cases or parts of the code which are not directly managed within the setup of the (in our case) MIFARE Classic card and reader IC CLRC663.

For example:

This example shows how the function ***phhalHw_Rc663_SetConfig()*** could be reduced.

We focus on the switch - case “PHHAL_HW_CONFIG_RXDATARATE”. Inside this condition there is a Felica card parameters setting, but Felica is not used in accordance to the project setup. These parts are excluded from compilation by the SCALE_DOWN_RC663_EXTENDED define. See the following code correction.

```
            case PHHAL_HW_CONFIG_RXDATARATE:

    #ifndef SCALE_DOWN_RC663_EXTENDED
                /* Felica card -> TxDataRate eqals the new RxDataRate */
                if (pDataParams->bCardType == PHHAL_HW_CARDTYPE_FELICA)
                    {
                    wDataRate = wValue;
                    }
                /* Other Cards -> read TxDataRate from shadow */
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

```
            else
                {
                wDataRate = pDataParams->wCfgShadow[PHHAL_HW_CONFIG_TXDATARATE];
                }
#else
            wDataRate = pDataParams->wCfgShadow[PHHAL_HW_CONFIG_TXDATARATE];
#endif

            /* Evaluate hardware settings */
#ifdef SCALE_DOWN_RC663
            statusTmp = ( phhalHw_Rc663_SetCardMode(
#else
            PH_CHECK_SUCCESS_FCT(statusTmp, phhalHw_Rc663_SetCardMode(
#endif
                    pDataParams,
                    wDataRate,
                    wValue,
                    pDataParams->wCfgShadow[PHHAL_HW_CONFIG_SUBCARRIER]));

#ifndef SCALE_DOWN_RC663_EXTENDED
            /* Felica card -> Update TxDataRate in shadow*/
            if (pDataParams->bCardType == PHHAL_HW_CARDTYPE_FELICA)
                {
                pDataParams->wCfgShadow[PHHAL_HW_CONFIG_TXDATARATE] = wDataRate;
                }
#endif

            /* Write config data into shadow */
            pDataParams->wCfgShadow[wConfig] = wValue;
            break;
```

Example continuation:

We focus on the next switch - case “PHHAL_HW_CONFIG_SERIAL_BITRATE” in the function ***phhalHw_Rc663_SetConfig()***. This one is focused to set the bit rate of the UART interface. The UART interface is not used in our example and therefore this switch - case could be fully excluded from compilation by the SCALE_DOWN_RC663_EXTENDED define. See the following code correction.

```
#ifndef SCALE_DOWN_RC663_EXTENDED
        case PHHAL_HW_CONFIG_SERIAL_BITRATE:

            switch (wValue)
                {
                case PHHAL_HW_RS232_BITRATE_7200:
                    bRegister = PHHAL_HW_RC663_SERIALSPEED_7200;
                    break;
                case PHHAL_HW_RS232_BITRATE_9600:
                    bRegister = PHHAL_HW_RC663_SERIALSPEED_9600;
                    break;
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

```
                case PHHAL_HW_RS232_BITRATE_1228800:
                    bRegister = PHHAL_HW_RC663_SERIALSPEED_1228800;
                    break;
                default:
                    return PH_ADD_COMPCODE(PH_ERR_INVALID_PARAMETER,
PH_COMP_HAL);
                }

            /* Set the register value */
#ifdef SCALE_DOWN_RC663
            statusTmp = ( phhalHw_Rc663_WriteRegister(pDataParams,
PHHAL_HW_RC663_REG_SERIALSPEED, bRegister));
#else
            PH_CHECK_SUCCESS_FCT(statusTmp, phhalHw_WriteRegister(pDataParams,
PHHAL_HW_RC663_REG_SERIALSPEED, bRegister));
#endif
            break;
#endif
```

4.4 Reduction of cross connections

After the corrections from above, we continue working with the module phalMfc_Sw.c instead of the module phalMfc.c. Functions from this module call functions from the PAL and those call functions from the HAL. These cross connections can be removed and called functions from the PAL could be replaced by HAL functions directly.

This correction requires a sensitive approach because input parameters of AL functions cannot respond to input parameters of HAL functions.

For example:

There is a typical function where input parameters change is necessary.

Function *phalMfc_Authenticate()* is called from the main as follows:

```
status = phalMfc_Authenticate(&alMfc, 0, PHHAL_HW_MFC_KEYA, 0, 0, bUid, bLength);
```

This function can be replaced according to3.5 as follows:

```
status = phalMfc_Sw_Authenticate(&alMfc, 0, PHHAL_HW_MFC_KEYA,0, 0, bUid, bLength);
```

Internal functions can be changed like this. It is necessary to reconnect input data structure mapping from application data parameters to protocol data parameters using temporary pointer *pDataParams_temp* remapping.

```
phStatus_t phalMfc_Sw_Authenticate(
        phalMfc_Sw_DataParams_t * pDataParams,
        uint8_t bBlockNo,
        uint8_t bKeyType,
        uint16_t wKeyNo,
        uint16_t wKeyVersion,
        uint8_t * pUid,
        uint8_t bUidLength
)
    {
```


All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

```
     .. .. ..

    phpalMifare_Sw_DataParams_t * pDataParams_temp;
    pDataParams_temp = (phpalMifare_Sw_DataParams_t *) pDataParams-
>pPalMifareDataParams;

    /* check if software key store is available. */
    if (pDataParams->pKeyStoreDataParams == NULL)
        {
        /* There is no software keystore available. */
//        return phpalMifare_MfcAuthenticateKeyNo
//                pDataParams->pPalMifareDataParams,
//                bBlockNo,
//                bKeyType,
//                wKeyNo,
//                wKeyVersion,
//                &pUid[bUidLength - 4]);


        return phhalHw_Rc663_MfcAuthenticateKeyNo(
                pDataParams_temp->pHalDataParams,
                bBlockNo,
                bKeyType,
                wKeyNo,
                wKeyVersion,
                &pUid[bUidLength - 4]);
        }
```

4.5 Individual corrections

Other corrections can be verified individual and performed very strictly based on real user application, and hardware usage and its configuration.

Rule 1:

Focus on the biggest code size functions. These could be the following:

phhalHw_Rc663_SetConfig()

phhalHw_Rc663_ApplyProtocolSettings()

phhalHw_Rc663_GetConfig()

phhalHw_Rc663_Exchange()

phhalHw_Rc663_SetConfig_Int()

phhalHw_Rc663_SetCardMode()

phhalHw_Rc663_GetFdt()

phhalHw_Rc663_Command_Int()

phhalHw_Rc663_WaitIrq()

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

In general, the code found in the functions listed above is not necessary for closed user application. They are either only used for debugging purposes during development, or can be replaced by hardcoded settings.

For example:

In the function *phhalHw_Rc663_Exchange()* the initial parameters check can be removed if the input parameters are filled in by known values. The following source code can be removed.

```
phStatus_t phhalHw_Rc663_Exchange(
        phhalHw_Rc663_DataParams_t * pDataParams,
        uint16_t wOption,
        uint8_t * pTxBuffer,
        uint16_t wTxLength,
        uint8_t ** ppRxBuffer,
        uint16_t * pRxLength
)
    {
    phStatus_t  PH_MEMLOC_REM status;
    phStatus_t  PH_MEMLOC_REM statusTmp;
    uint16_t    PH_MEMLOC_REM wNumPrecachedBytes;
    uint16_t    PH_MEMLOC_REM wFifoBytes;
    uint8_t     PH_MEMLOC_REM bIrq0WaitFor;
    uint8_t     PH_MEMLOC_REM bIrq1WaitFor;
    uint8_t     PH_MEMLOC_REM bIrq0Reg = 0x00;
    uint8_t     PH_MEMLOC_REM bIrq1Reg = 0x00;
    uint8_t     PH_MEMLOC_REM bRegister;
    uint8_t     PH_MEMLOC_REM bError;
    uint8_t     PH_MEMLOC_REM bNoData;
    uint32_t    PH_MEMLOC_REM dwTimingSingle;
    uint8_t *   PH_MEMLOC_REM pTmpBuffer;
    uint16_t    PH_MEMLOC_REM wTmpBufferLen;
    uint16_t    PH_MEMLOC_REM wTmpBufferSize;

#ifndef SCALE_DOWN_RC663_EXTENDED
    /* Check options */
    if (wOption & (uint16_t)~(uint16_t)(PH_EXCHANGE_BUFFERED_BIT |
PH_EXCHANGE_LEAVE_BUFFER_BIT))
        {
        return PH_ADD_COMPCODE(PH_ERR_INVALID_PARAMETER, PH_COMP_HAL);
        }
#endif // SCALE_DOWN_RC663_EXTENDED
```

In the same function the condition of about communication interface could be reduced to the used type.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

```
    /* Fill the global TxBuffer */
    /* Note: We always need to buffer for SPI, else the input buffer would get
overwritten! */
#ifndef SCALE_DOWN_RC663_EXTENDED
    if ((wOption & PH_EXCHANGE_BUFFERED_BIT) ||
            (pDataParams->bBalConnectionType == PHHAL_HW_BAL_CONNECTION_SPI) ||
            (pDataParams->bBalConnectionType == PHHAL_HW_BAL_CONNECTION_I2C))
#else
   if ((wOption & PH_EXCHANGE_BUFFERED_BIT) ||
           (pDataParams->bBalConnectionType == PHHAL_HW_BAL_CONNECTION_SPI))
#endif // SCALE_DOWN_RC663_EXTENDED
        {
        /* retrieve transmit buffer */
#ifdef SCALE_DOWN_RC663
        statusTmp = ( phhalHw_Rc663_GetTxBuffer(pDataParams, PH_ON, &pTmpBuffer,
&wTmpBufferLen, &wTmpBufferSize));
#else
        PH_CHECK_SUCCESS_FCT(statusTmp, phhalHw_Rc663_GetTxBuffer(pDataParams,
PH_ON, &pTmpBuffer, &wTmpBufferLen, &wTmpBufferSize));
#endif

        if (wTxLength != 0)
```

Rule 2:

A similar procedure could apply to the smallest code size functions. The body of the function could be copied directly to the source code in the applied place. However, the operation can only be realized if the function is one of those used in the following application:

Function *Fill_Block()* is used for fill in data field.

```
/*============================================================================
 * FUNCTION:    Fill_Block
 *
 * Description:
 *      copy the key to the field for comparation
 *
 ----------------------------------------------------------------------------*/
#ifndef SCALE_DOWN_RC663
static void Fill_Block (uint8_t *pBlock, uint8_t MaxNr)
    {
    uint8_t i;

    for (i = 0; i <= MaxNr; i++)
        {
        *pBlock++ = i;
        }
    }
#endif
```

The replacement in the *main()* could be as follows:

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

```
#ifndef SCALE_DOWN_RC663_EXTENDED
                    /* fill block with data */
                    Fill_Block(bBufferReader, 15);
#else
                    for (i = 0; i <= 15; i++)
                        {
                        bBufferReader[i] = i;
                        }
#endif // SCALE_DOWN_RC663_EXTENDED
```

4.6 Final compilation and FLASH saving

The build setup of this project example (2.3) is set in the file *…/intfs/phhalHw.h*.

The conditions have to be defined as follows:

```
#define SCALE_DOWN_RC663
#ifdef SCALE_DOWN_RC663
#define SCALE_DOWN_RC663_EXTENDED
#endif
```

Another necessary correction in the changed modules is to add the file *phhalHw.h* to the includes. (NOTE: this correction is already done for this project example)

Finally the reduced code size after compiling the example is 33.24%

The code size has been reduced from 0x4492 (17554 bytes) to 0x2dc6 (11718 bytes).

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

5. Build configuration

5.1 Release configuration

The release configuration commonly reduces the application code to the minimum memory size. The Tool Settings of the IDE are set as follows:

Symbols

NDEBUG	- no debug messages
__DISABLE_WATCHDOG	- watchdog module is allowed for release configuration and for the LPC1227.

Basic symbols are commonly added.

Optimization

-O2	this optimization also turns on the –Os which is optimization for size.

Debugging

None	the minimal debug support
-g0	produces no debug information at all

For more configurations see [3] chapters 6.2. and 6.3.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

6. References

[1] LPCXpresso Code Red website
www.nxp.com/redirect/lpcxpresso.code-red-tech.com/LPCXpresso

[2] LPCXpresso NXP website
www.nxp.com/redirect/lpcware.com/lpcxpresso

[3] software example of RC663-LPC1227-Classic
http://www.nxp.com/demoboard/PREV601.html#documentation

[4] embedded artists – general site
www.nxp.com/redirect/embeddedartists.com/products/lpcxpresso

[5] embedded artists – LPC1227 board description
www.nxp.com/redirect/embeddedartists.com/products/lpcxpresso/lpc1227_xpr.php

[6] blueboard CLEV663B
http://www.nxp.com/demoboard/CLEV663B.html

[7] **RC663 data sheet**
http://www.nxp.com/documents/data_sheet/CLRC663.pdf

[8] **LPC1227 User Manual**
http://www.nxp.com/documents/user_manual/UM10441.pdf

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

7. Legal information

7.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

7.3 Licenses

Purchase of NXP ICs with ISO/IEC 14443 type B functionality

This NXP Semiconductors IC is ISO/IEC 14443 Type B software enabled and is licensed under Innovatron's Contactless Card patents license for ISO/IEC 14443 B.

The license includes the right to use the IC in systems and/or end-user equipment.

RATP/Innovatron Technology

7.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

MIFARE — is a trademark of NXP B.V.

MIFARE DESFire — is a trademark of NXP B.V.

MIFARE Plus — is a trademark of NXP B.V.

MIFARE Ultralight — is a trademark of NXP B.V.

ICODE — is a trademark of NXP B.V.

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2013. All rights reserved.

8. Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

© NXP B.V. 2013. **All rights reserved.**

For more information, visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 11 March 2013
259610
Document identifier: AN11342