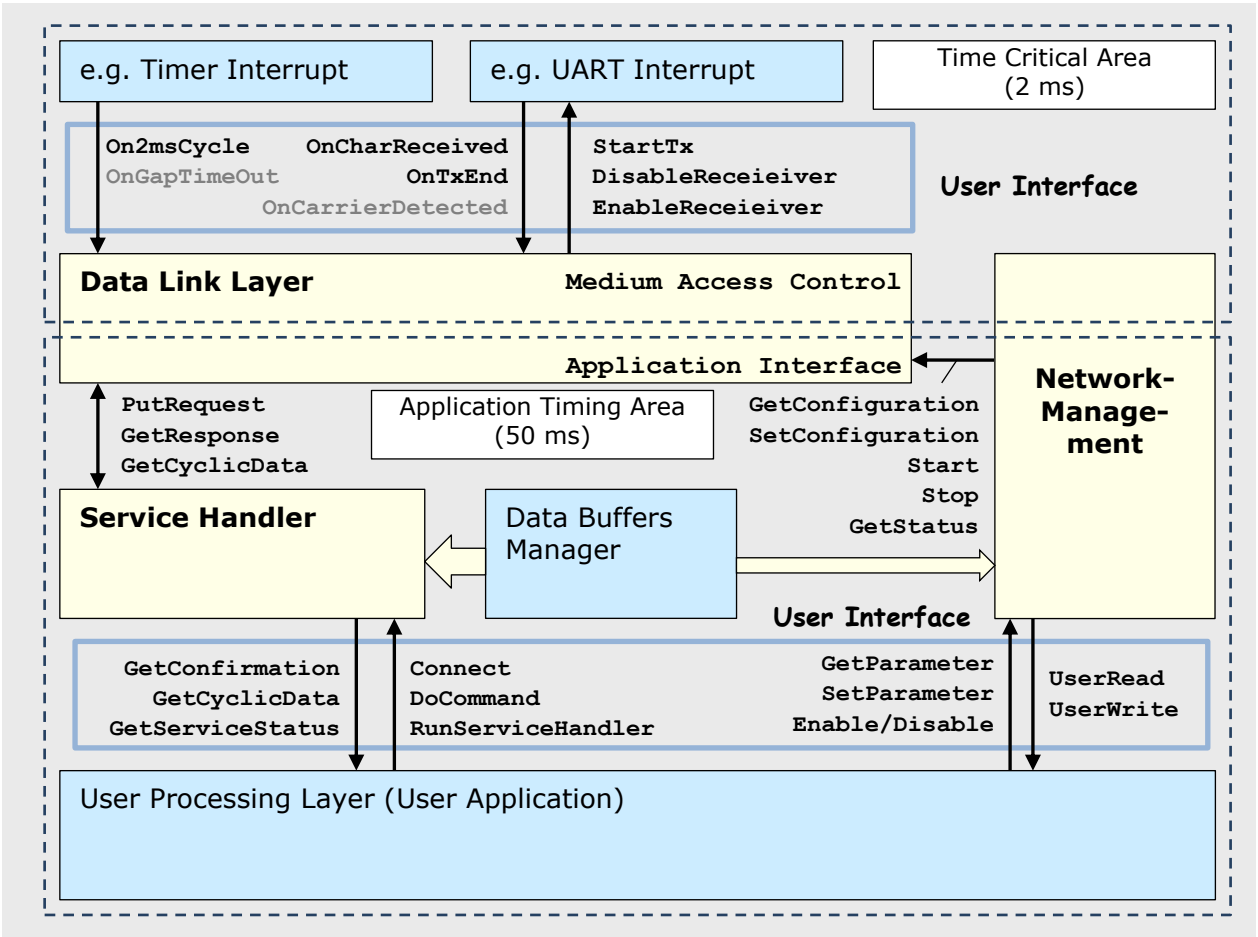
Borst Automation Embedded Solutions
Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

Hart Master C++ 8.0E

Source Code Module

The package includes all modules needed to represent the Hart protocol for a master. The package is written in standard C++ and does not use a direct connection to a system environment. Data link layer, application layer and network management of the Hart protocol are implemented. The connection to the outside occurs via three interfaces: OSAL, USER and HAL. Special properties are:

- No external dynamic memory management. The amount of reserved RAM remains constant.
- The number of objects is determined at compile time and startup.
- No operating system is required to integrate the software. Timers and serial interrupts are enough.
- The user interface (USER) is very close to the interface of the existing Hart DLL in HartTools 7.6.
- With the exception of the timer and the UART interrupt, there are no callbacks to registered functions.



Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

Contents

Overview

Details for the Hart Protocol are provided via the following link: https://www.fieldcommgroup.org/technologies/hart.

Implementation Considerations

Microcontrollers which are used today for HART devices are at least 16 Bit microcontrollers. Otherwise the complexity of the measurement and number of parameters could not be managed.

☞ Low amount of memory.

The amount of memory is always critical because software kind of behaves like an ideal gas. It uses to fill the given space. Nevertheless, the coding of the Hart Master was done as carefully as possible regarding the amount of flash memory and RAM.

☞ The user needs source code.

The Hart Protocol requires a strict timing specially for burst mode support and the primary and secondary master time slots. To provide the optimum transparency to the user to allow all kinds of debugging and to give the opportunity to optimize code in critical sections, the Hart Master Software is not realized as a library but delivered as source code.

HAL (Hardware Abstraction Layer)

☞ OSAL is including the HAL.

A Hardware Abstraction Layer is needed to design the interface of a software component independent from the hardware platform. In this very small interface of the Hart master a distinction of HAL and OSAL was not made. Therefore only an

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

Operating System Abstraction Layer is defined which is covering all the needs of an appropriate HAL.

Architecture

You can find a graphic representation of the architecture on the cover page of this datasheet.

The software is mainly divided into two areas. One is the time critical part, which is needed to meet the requirements of the time controlled dual master protocol of Hart. The other is the area were the application software is working, which is far less time critical.

☞ The software architecture is optimized for systems with very few resources.

The figure above is clearly showing also two user interfaces. There is a user interface which is connecting the Hart Master software to a timer control interrupt and a UART interrupt which are used for the 'fast' service procedures. Most of the Hart protocol functionalities are solved in the timer part, which may run on interrupt level. There are arguments for and against this kind of implementation but you ever end up at a point that the incoming frame has to be processed as quickly as possible. So why not spending a few microseconds more once the program has already reached the interrupt level. The Hart protocol is not very complex but it needs to be processed fast enough to catch a precise timing.

The load produced by the implementation is not very high. Because the communication runs with a speed of 1200 bit/s usually there is nothing to do in the 1 ms cycle than to keep track of the timing. Only every 10 ms - if a frame is coming in - a character has to be processed. The processing is done in an incremental way thus not implying the execution of too much instructions.

The split between the time critical area and the user application is done within the Data Link Layer and the so called Network Management. However, the user have not to take any special on these separations except the provision of a few OSAL services for Locking out other tasks. There is an 'atomic' lock out level which has to lock out the interrupts of the Data Link Layer as well as concurrent processes. The other level is 'critical section' which is locking out concurrent processes. More details are described in another chapter of this document.

☞ The top level user interface is communication independent.

The interface to the user's application is located on top of the User Data Processing Layer (User Application). Functions with names starting with 'User' are function which are expected by the communication stack to be provided by the user. Another set of functions are called by the user's software on demand. There is no restriction when these functions may be called. The functions for the user are neutral and does not show that they are used for HART communications.

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

There are a few data objects which are required for Hart protocol and which may be set by the user or an external Hart master. These are such as the tag name and the address. If e.g. the address of the Hart slave is changed through the network the Network-Management will call the user layer to store the data in the NV-memory. If the address is changed through the local HMI of the Hart device, the user layer calls Network-Management of Hart to advise the Data Link Layer protocol to work with the new address. The function used for this setting is SetParameter.

In the above figure the parts of the Hart Master 7.6 are shown in yellow color while the user parts are marked with blue. A major part is the block called User Data and Command Description. This block is binding the user data parts to the Hart commands and or function provided by the user. This is finally a set of tables stored in the flash memory of the device.

☞ The Data Link Layer is an independent piece of software.

The figure is also showing a set of functions between the command interpreter the network management and the data link layer. These functions may be used if the developer decides to use only the data link layer by providing its own command interpreter and network management.

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

Public Functions

Category	Name	Description
Data Link Layer		
DLL Initialization	**Init**	Initializes the protocol layer. This has to be the first call into the Hart Master Data Link Layer.
DLL Operation	**GetConfiguration**	Returns the configuration from the protocol layer.
	SetConfiguration	Sets the configuration for the protocol layer.
	Start	Starts the Data Link Layer protocol, enables the receiver.
	Stop	Stops the Data Link Layer protocol. disables the receiver.
	GetStatus	Return the status of the Data Link Layer.
DLL MAC Access User Interface	**On2msCycle**	This routine has to be called on every two milliseconds. It run the HART protocol including mechanism for sending automated responses and the handling of burst frames. May be called by interrupt.
	OnCharReceived	Has to be called if a character was received. May be called from interrupt.
	OnTxEnd	Has to be called if transmitting octets is completed. This call has to be done after the stop bit of the last byte of a stream was sent. The function may be called from interrupt.
	StartTx	A request to the user part of the software for starting the transmission of a byte stream. This function is called from inside On2msCycle.
	DisableReceiver	A request to the user part of the software for switching off the receiver. This function is called from inside On2msCycle.
	EnableReceiver	A request to the user part of the software for switching on the receiver. This function is called from inside On2msCycle.
	OnGapTimeout	This function call is optional. If the user has enough timer resources he may implement his own more precise Gap Time Out. This would allow more jitter on the 2 ms cycle.
	OnCarrierDetected	This function call is optional. If the user's hardware is in able to detect the carrier this can be used as a better indication for frame start and frame end than the gap time out.
DLL APP Iface	**GetRequestData**	Returns the latest incoming request (usually a Hart command). This function is called if the user software has time.
	SetResponseData	Called from the user software (the command interpreter) to place the response to a command.
	SetCyclicData	Called from the user software. Used to set cyclic data for the burst frames. The Data Link Layer is managing to send the data if the protocol is allowing or requiring it.

Table 1: List of Data Link Layer Functions

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

User Data Processing Layer (User Application)		
Command Interpreter	**Read**	This function is called from the Hart Master software to read a parameter which is described in the User Data and Command Description.
	Write	This function is called from the Hart Master software to write a parameter which is described in the User Data and Command Description.
	HandleCommand	This function is called from the Hart Master if a command was received, which could not be handled by the Hart Master service handler.
	PutCyclicData	The user layer calls this function to update the cyclic data (e.g. measurement values).
	PutDeviceStatus	The user layer calls this function whenever the device status has changed.
	RunServiceHandler	This function is used to execute the service handler from application level. It has to be called in a cycle of about 50 ms .. 200 ms.
Network Management	**Read**	See description above.
	Write	See description above.
	SetParameter	This functions is provided for the user layer to provide settings like slave address or tag name.
	GetParameter	This function is provided for the user layer to get settings.
	Enable	Enables or disables the Hart Protocol in the device.
	Init	Initializes the master stack including the command handler.

Table 2: List of User Layer Functions

System Requirements

It is difficult to estimate the system requirements for targets based on different micro controllers and different development environments. The following is therefore giving a very rough scenario for the target system resources.

Item	Requirement/Size	Comment
RAM	**64k**	Depends very much on addressing structure of the controller and the used compiler and linker.
ROM (Flash)	**100k**	
Timing	**2 ms Timer interrupt**	2 ms is the minimum requirement, 1 ms would be much better.
	50 ms cyclic all from task level	This is needed to run the command interpreter.
I/O	**UART and Hart MODEM Rx and Tx functions**	Carrier detection would be helpful but is not required.
System	**Simple math +-*/** **memcpy()** **memset()** **memcmp()**	Only a few standard library functions are required. There is no special need for multi tasking, messaging or semaphores.
	1 ms timing resolution	

Table 3: Embedded System Requirements

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

Prerequisites

The 'normal' project under VS 2019 can be used immediately. However, a GNU compiler is required for the make projects.

Visual Studio 2019

I used the following version of Visual Studio 2019:

```
Microsoft Visual Studio Professional 2019
Version 16.11.14
```

Further installations for VS 2019 are not required.

Visual Studio Code

I used the following version of Visual Studio 2019:

```
Version: 1.76.2 (user setup)
Commit: ee2b180d582a7f601fa6ecfdad8d9fd269ab1884
```

The following extensions are required: tbd.

Implementation Targets

Windows Computer

Tbd.

Migration to Linux

I think, porting to a Raspberry Pi 4 would be a good idea for this purpose. Of course, the MainLoop module must then be further abstracted to the Linux console, and two more files are added to the special files marked 'Win32'.

These new files will be marked as 'Linux'.

Implementation on nRF52840

Finally, porting to an nRF52840 would show whether the concept delivers what it promises.

A serial interface would be used as the connection, which could be replaced by a Bluetooth connection in a further modification.

Walter Borst
Kapitaen-Alexander-Strasse 39
27472 Cuxhaven, GERMANY
Fon:+49 (0) 4721 6985100
E-Mail: walter.borst@borst-automation.de
Home: https://www.borst-automation.de/

Appendix

Abbreviations

Abbreviation	Description
HCF	Hart Communication Foundation
DLL	Windows: Dynamic Link Library OSI-ISO: Data Link Layer
HAL	Hardware Abstraction Layer
HART	Highway Addressable Remote Transducer See also: http://en.wikipedia.org/wiki/Highway_Addressable_Remote_Transducer_Protocol
HMI	Human Machine Interface
ISO	International Standards Organisation
MODEM	MOdulator DEModulator
NV-memory	Non-Volatile memory
OSAL	Operating System Abstraction Layer
OSI	Open Systems Interconnection
UART	Universal Asynchronous Receiver Transmitter