USER MANUAL

Accessory 72EX



UMAC Fieldbus Interface

3x0-603958-xUxx

September 1, 2015



Single Source Machine Control Power // Flexibility // Ease of Use 21314 Lassen Street Chatsworth, CA 91311 // Tel. (818) 998-2095 Fax. (818) 998-7807 // www.deltatau.com

Copyright Information

© 2015 Delta Tau Data Systems, Inc. All rights reserved.

This document is furnished for the customers of Delta Tau Data Systems, Inc. Other uses are unauthorized without written permission of Delta Tau Data Systems, Inc. Information contained in this manual may be updated from time-to-time due to product improvements, etc., and may not conform in every respect to former issues.

To report errors or inconsistencies, call or email:

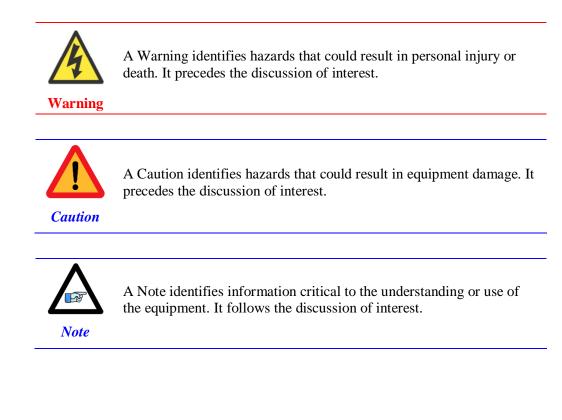
Delta Tau Data Systems, Inc. Technical Support

Phone: (818) 717-5656 Fax: (818) 998-7807 Email: <u>support@deltatau.com</u> Website: <u>http://www.deltatau.com</u>

Operating Conditions

All Delta Tau Data Systems, Inc. motion controller products, accessories, and amplifiers contain static sensitive components that can be damaged by incorrect handling. When installing or handling Delta Tau Data Systems, Inc. products, avoid contact with highly insulated materials. Only qualified personnel should be allowed to handle this equipment.

In the case of industrial applications, we expect our products to be protected from hazardous or conductive materials and/or environments that could cause harm to the controller by damaging components or causing electrical shorts. When our products are used in an industrial environment, install them into an industrial electrical cabinet or industrial PC to protect them from excessive or corrosive moisture, abnormal ambient temperatures, and conductive materials. If Delta Tau Data Systems, Inc. products are directly exposed to hazardous or conductive materials and/or environments, we cannot guarantee their operation.



REV	DESCRIPTION	DATE	CHG	APPVD
1	Preliminary Manual	11/05/12	Sina Sattari	SS
2	Added Power PMAC support and address settings based upon 603958-102	09/24/13	Sina Sattari	SS
3	Corrected ACC72EX.Data8[i] references	10/21/2013	Sina Sattari	SS
4	Added C code and setup examples; corrected typos	7/29/15	DCDP	SS

Table of Contents

INTRODUCTION	8
THEORY OF OPERATION	9
UBUS Interface	9
How ACC-72EX Works	9
Turbo PMAC Memory	10
Power PMAC Memory	
Hilscher ComX Module Addressing to Turbo PMAC Addressing Conversion	
Hilscher ComX Module Addressing to Power PMAC Addressing Conversion	
HARDWARE	14
E3: UBUS Address	
CS16- Identification	
Identification Information	
Jumper Settings	15
Option Identification Jumpers	
E-Point Jumper Settings	
Communication Option-Dependent E-Point Jumper Settings	
Connector Pinouts	16
Fieldbus Port (J4)	16
Real-time Ethernet Ports (Ethernet 0 & Ethernet 1)	16
Diagnostics Port (Micro A USB)	
DPRAM MEMORY MAP	17
DPRAM Blocks	18
System Channel	18
Handshake Channel	20
Communication Channel	
Application Channel	34
Auto-Generated Dual-Ported Memory Map	34
Address Converter	34
Memory Map Generator	35
DPRAM DATA PROCESSING	50
Non-Cyclic Data Exchange	50
Message or Packets	50
About System and Channel Mailbox	53
Command and Acknowledge	
Using ulSrc and ulSrcId	
How to Route rcX Packets	
Client/Server Mechanism	
Input/Output Data Image	59

Process Data Handshake Modes	59
Start / Stop Communication	
Controlled or Automatic Start	62
Start / Stop Communication through Dual-Port Memory	62
Reset Command	63
System Reset vs. Channel Initialization	63
Resetting netX through Dual-Port Memory	63
System Reset through Packets	66
SOFTWARE SETUP	67
Required Software Packages	67
SyCon.NET Software Setup	
ACC-72EX Setup Assistant	
Turbo PMAC Setup for Using ACC-72EX	
Initialization PLC	
Watchdog Function	
Enabling the Communication Bus	
Locating the Input/Output Data Image in PMAC	
Reading/Writing from/to Input/Output Data Images	
Power PMAC Setup for Using ACC-72EX	
ACC72EX[<i>i</i>]. Non-Saved Data Structures	
C Programming Access to ACC-72EX Structures	
Global Header for Power PMAC Projects	
Initialization PLC	
Startup	
Watchdog Function	
Enabling the Communication Bus	
Locating the Input/Output Data Image in PMAC	99
DIAGNOSTICS	
LEDs	
PROFIBUS-DP – Master – OPT10	
PROFIBUS-DP – Slave – OPT11	101
DeviceNet – Master – OPT20	
DeviceNet – Slave – OPT21	
CANopen – Master – OPT30	
CANopen – Slave – OPT31	103
CC-Link – Slave – OPT51	
EtherCAT – Master – OPT60	
EtherCAT – Slave – OPT61	
EtherNet/IP – Scanner/Master – OPT70	
EtherNet/IP – Adaptor/Slave – OPT71	
Open Modbus/TCP – OPT80	108

PROFINET IO – Controller – OPT90	
PROFINET IO – Device – OPT91	110
APPENDIX A – SETUP EXAMPLES	
SYCON.net Setup	111
RSLogix 5000 Setup	
COMX Test PLC	
APPENDIX B – TURBO PMAC MEMORY MAPS	136
APPENDIX C – POWER PMAC MEMORY MAPS	

INTRODUCTION

This manual provides the information needed to configure ACC-72EX, a fieldbus/real-time Ethernet interface for the Turbo or Power UMAC. The ACC-72EX is equipped with a "gateway" daughter card that allows the UMAC (also referred to as host application) to send and receive data through the supported fieldbus/real-time Ethernet protocols. The gateway used is the COMX CN series manufactured by the Hilscher Corporation. Relevant hyperlinks are provided in Appendix D for in-depth information regarding these modules.

There are three connectors located on the front of the ACC-72EX:

First, a Micro B USB connector, which is specified as "Diagnostic Port," and provides USB connectivity to Hilscher's "SyCon.NET" software.

The second connector, which is referred to as the "Fieldbus Port", is a 9-Pin Male D-Sub connector which is used for connecting the fieldbus link to ACC-72EX. The fieldbus protocols supported through this port are:

- PROFIBUS-DP Master OPT10
- PROFIBUS-DP Slave OPT11
- DeviceNet Master OPT20
- DeviceNet Slave OPT21
- CANopen Master OPT30
- CANopen Slave OPT31
- CC-Link Slave OPT51

The third connector is composed of two RJ-45 ports which provide connection to real-time Ethernet networks. The following real-time Ethernet protocols are supported through these ports:

- EtherCAT Master OPT60
- EtherCAT Slave OPT61
- EtherNet/IP Scanner/Master OPT70
- EtherNet/IP Adaptor/Slave OPT71
- Open Modbus/TCP OPT80
- PROFINET IO Controller OPT90
- PROFINET IO Device OPT91

The protocol is dependent upon the equipped COMX gateway. The hardware cannot be programmed for an alternate protocol or change from slave to master or vice versa. However, should the COMX gateway be replaced with one supporting another protocol, the baseboard would function properly as a communications link to UMAC. In this case, proper jumper settings should be set up to ensure proper functionality on communication lines and option detection.

Most gateway cards get their power from the UBUS back plane; however, the DeviceNet option (Options 3 & 4) requires an external 24 VDC power supply through the "Fieldbus Port."

THEORY OF OPERATION

The ACC-72EX board is organized as a motherboard/daughter board system. The motherboard contains the UBUS interface, diagnostics, and the fieldbus connections. The daughter board contains the intelligence (firmware which will be referred as netX) and the interface electronics required for each fieldbus. There is a different daughter board for each fieldbus.

The netX firmware on the daughter board implements each fieldbus communications protocol. Fieldbus data is transferred to/from the fieldbus and placed in a Dual-Ported RAM (memory) on the daughter board. The structure of this DPRAM is given later in this manual and is common for all the field buses. ACC-72EX supports up to 64K DPRAM on each device (one full chip-select width).

The PMAC side of the DPRAM is interfaced to the UBUS. PMAC programs access the fieldbus data by reading or writing data to memory addresses corresponding to the location of the PMAC Gateway 3U board's DPRAM.

UBUS Interface

The UBUS is Delta Tau's bus interface for the UMAC controller. The ACC-72EX maps to the UBUS as a DPRAM style board. It occupies contiguous memory locations (both X and Y memory for Turbo PMAC) of the lower two bytes of the 24-bit (middle 16 bits of each 32 bit word for Power PMAC), DPRAM addresses. Because the DPRAM size supported on ACC-72EX can be as large as 64K, each card will occupy one full Chip Select addressing space. There can be a maximum of two ACC-72EX cards per Turbo/Power UMAC (cannot be in a MACRO Station).

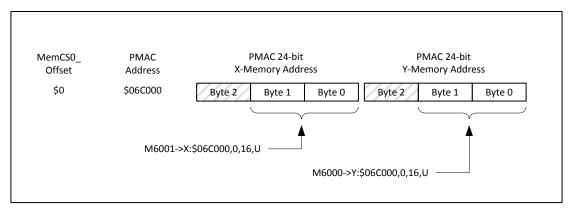
M-Variables can be mapped to these locations to move data to and from the fieldbus and PMAC. In addition to fieldbus data, there is a block of memory that indicates the ACC-72EX's status.

How ACC-72EX Works

- 1. The ACC-72EX organizes fieldbus bytes in dual-port memory on the COMX module. These fieldbus bytes are mapped into PMAC's memory space via the UBUS interface.
- 2. PMAC M-Variables are used to move data to and from the fieldbus or to control the COMX board.
- 3. An E-point jumper on the ACC-72EX sets the address of the board in PMAC memory space.
- 4. The COMX board is configurable via a USB port. SYCON.NET is provided with the COMX board for this purpose.
- 5. Diagnostic LEDs are provided for a visual indication of board status.

Turbo PMAC Memory

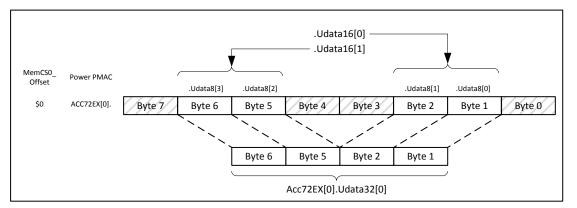
Turbo PMAC uses a DSP (Digital Signal Processor) with a 24-Bit architecture that uses two memory areas: Y and X Memory. Memory is accessed in PMAC programs using M-Variables. The definition of an M-Variable includes its number, address, offset, width, and type. Refer to the Turbo PMAC Software Reference Manual or Turbo PMAC User Manual for additional explanation of M-Variables and their specification, such as in the "M-Variables" section in the User manual.



Turbo PMAC Memory Organization

Power PMAC Memory

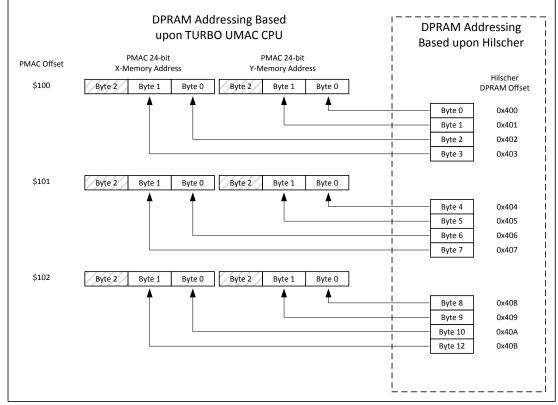
Power PMAC uses 32-bit data bus architecture. ACC-72EX Memory is accessed in Power PMAC data structures or their equivalent **#define** statements. The **#define** statements are included later in this manual.



Power PMAC Memory Organization

Hilscher ComX Module Addressing to Turbo PMAC Addressing Conversion

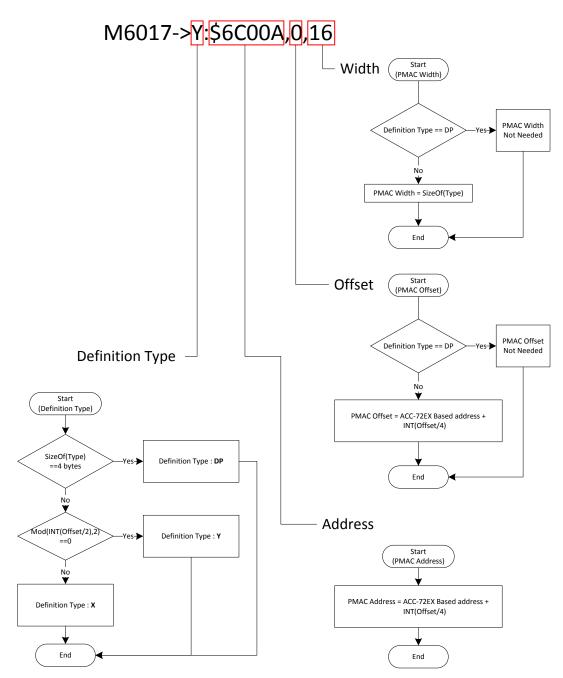
As explained in previous sections, Turbo PMAC places 4 bytes of Hilscher ComX memory data in each PMAC memory word. This means that for each address offset increment on the PMAC side, there will be 4 increments of offset addresses on the Hilscher DPRAM side. The following example shows PMAC addressing for equivalent offset addresses of 0x400 in Hilscher documentation.



Consumed Data Flow

In general, the following flowcharts can be used to convert any Hilscher DPRAM addressing to PMAC's addressing format:

Sy	System Information Block							
Offset Type Name				Description				
0x	(0028	UINT16	usDeviceClass	Device Class netX Device Class (see page 34)				



An address conversion tool is provided in the ACC-72EX Setup Assistant software.

Hilscher ComX Module Addressing to Power PMAC Addressing Conversion

In Power PMAC, specific **Acc72EX**[*i*] data structures have been implemented which allow bit-wide, byte-wide, 2-byte and 4-byte access to Hilscher ComX Dual Ported RAM.

Acc72EX[*i*].Udata16[*j*] structures can be used for individual bit access, both for read and for write purpose.

DPRAM Addressing Based upon Power UMAC CPU	DPRAM Addressing Based upon Hilscher
Power PMAC Structures	Hilscher DPRAM Offset
Acc72EX[0].Udata32[256] Acc72EX[0].Udata32[256] Acc72EX[0].Udata16[512] Acc72EX[0].Udata8[1025] Acc72EX[0].Udata8[1026] Acc72EX[0].Udata8[1027] Byte 1 Byte 1 Byte 2 Byte 3 Byte 3	Byte 0 0x400 (1024) Byte 1 0x401 (1025) Byte 2 0x402 (1026) Byte 3 0x403 (1027)
Acc72EX[0].Udata32[257] Acc72EX[0].Udata32[257] Acc72EX[0].Udata16[514] Acc72EX[0].Udata8[1028] Acc72EX[0].Udata8[1029] Acc72EX[0].Udata8[1029] Byte 4 Byte 5 Acc72EX[0].Udata8[1030] Acc72EX[0].Udata8[1031] Byte 7 I Byte	Byte 4 0x404 (1028) Byte 5 0x405 (1029) Byte 6 0x406 (1030) Byte 7 0x407 (1031)
Acc72EX[0].Udata32[258] Acc72EX[0].Udata32[258] Acc72EX[0].Udata16[516] Acc72EX[0].Udata8[1032] Acc72EX[0].Udata8[1033] Acc72EX[0].Udata8[1034] Acc72EX[0].Udata8[1035] Byte 10 Byte 12 Byte	Byte 8 0x408 (1032) Byte 9 0x409 (1033) Byte 10 0x40A (1034) Byte 12 0x40B (1035)

HARDWARE

E3: UBUS Address

E-point jumper E3 on the ACC-72EX controls the base address and range on the UBUS. Since each ACC-72EX uses full-13 bit addressing, it consumes all the memory addressable through each chip select. As a result, two is the maximum number of ACC-72EX boards that can be used in a Turbo UMAC rack.

E3	Turbo PMAC	Power PMAC
1-2	Y/X:\$6C000 - \$6FFFF	ACC-72EX[0] (\$E00000)
2-3	Y/X:\$74000 - \$7FFFF	ACC-72EX[1] (\$F00000)

The default location on Turbo PMAC is Y/X:\$6C000 - \$6FFFF (\$E00000 on Power PMAC).

Note:

Do not set the ACC-72EX to the DPR address range \$6C000-\$6FFFF if the UMAC is equipped with an Acc-54E. Acc-54E is set to this range as default.

CS16- Identification

One of the features of the UBUS is that memory locations, selected by CS16 (Chip Select 16/Active Low), were reserved for board identification information.

- Vendor ID (8 bits)
- Options Present (10 bits)
- Revision Number (4 bits)
- Product ID (14 bits)

This information (36 bits) is accessible directly with I-Variables added in Turbo PMAC Firmware 1.936 or later. A summary of the PMAC Gateway ID information is in the table below.

I39 controls the values reported.

I39=	I4942I4952 reports the following
0	36 bits (Vendor ID, Options present, Rev Number, Product ID)
1	8 bits (Vendor ID)
2	10 bits (Options Present) Reported by PMAC in HEX (\$)
3	4 bits (Revision Number)
4	14 bits (Product ID)
5	19 bits (Card Base Address)

Identification Information

The vendor ID, part number, and revision numbers are programmed into the ACC-72EX base board. The Option Number is set by jumpers on the board. The settings below are given for reference only. There is no need to change these from the factory settings.

T4	Comm. Desta del Ontion	Part Number	JP1	JP2	JP3	JP4	JP5	JP6	JP7	JP8	JP9
Item	Comm. Protocol Option	Part Number	Bit 9	Bit 10	Bit 11	Bit 12	Bit 13	Bit 14	Bit 15	Bit 16	Bit 17
1	PROFIBUS-DP – Master	310-603958-OPT	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF
2	PROFIBUS-DP – Slave	311-603958-OPT	ON	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF
3	DeviceNet – Master	320-603958-OPT	OFF	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF
4	DeviceNet – Slave	321-603958-OPT	ON	OFF	ON	OFF	ON	OFF	OFF	OFF	OFF
5	CANopen – Master	330-603958-OPT	OFF	ON	ON	ON	ON	OFF	OFF	OFF	OFF
6	CANopen – Slave	331-603958-OPT	ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF
7	CC-Link – Slave	351-603958-OPT	ON	ON	OFF	OFF	ON	ON	OFF	OFF	OFF
8	EtherCAT – Master	360-603958-OPT	OFF	OFF	ON	ON	ON	ON	OFF	OFF	OFF
9	EtherCAT – Slave	361-603958-OPT	ON	OFF	ON	ON	ON	ON	OFF	OFF	OFF
10	EtherNet/IP – Scanner/Master	370-603958-OPT	OFF	ON	ON	OFF	OFF	OFF	ON	OFF	OFF
11	EtherNet/IP – Adaptor/Slave	371-603958-OPT	ON	ON	ON	OFF	OFF	OFF	ON	OFF	OFF
12	Open Modbus/TCP	380-603958-OPT	OFF	OFF	OFF	OFF	ON	OFF	ON	OFF	OFF
13	PROFINET IO – Controller	390-603958-OPT	OFF	ON	OFF	ON	ON	OFF	ON	OFF	OFF
14	PROFINET IO – Device	391-603958-OPT	ON	ON	OFF	ON	ON	OFF	ON	OFF	OFF

Jumper Settings Option Identification Jumpers

E-Point Jumper Settings

Point	Default	Description
E1	1-2	Selection of Reset Polarity Signal for Hilscher Module:
		1-2 Selects Low True Reset
		2-3 Selects High True Reset
E2	2-3	Selection of UMAC CPU architecture. This selection affects the data bus and provides contiguous
		data addressing for the DPRAM:
		1-2 Power PMAC
		2-3 Turbo PMAC
E3	1-2	Selection of base address for ACC-72EX:
		1-2 Selects Y/X:\$6C000 - \$6FFFF (Acc72EX[0])
		2-3 Selects Y/X:\$74000 - \$7FFFF (Acc72EX[1])
E5	OFF	Connects DPRAM interrupt to UBUS IRQ-1
E6	OFF	Connects DPRAM interrupt to UBUS IRQ-2
E7	OFF	Connects DPRAM interrupt to UBUS IRQ-3

Communication Option-Dependent E-Point Jumper Settings

Comm. Protocol Option	Option Part Number	E8	E10	E11	E12
PROFIBUS-DP – Master	310-603958-OPT	1-2	OFF	OFF	OFF
PROFIBUS-DP – Slave	311-603958-OPT	1-2	OFF	OFF	OFF
DeviceNet – Master	320-603958-OPT	2-3	ON	OFF	OFF
DeviceNet – Slave	321-603958-OPT	2-3	ON	OFF	OFF
CANopen – Master	330-603958-OPT	OFF	OFF	ON	OFF
CANopen – Slave	331-603958-OPT	OFF	OFF	ON	OFF
CC-Link – Slave	351-603958-OPT	2-3	OFF	OFF	ON
EtherCAT – Master	360-603958-OPT	OFF	OFF	OFF	OFF
EtherCAT – Slave	361-603958-OPT	OFF	OFF	OFF	OFF
EtherNet/IP - Scanner/Master	370-603958-OPT	OFF	OFF	OFF	OFF
EtherNet/IP - Adaptor/Slave	371-603958-OPT	OFF	OFF	OFF	OFF
Open Modbus/TCP	380-603958-OPT	OFF	OFF	OFF	OFF
PROFINET IO – Controller	390-603958-OPT	OFF	OFF	OFF	OFF
PROFINET IO – Device	391-603958-OPT	OFF	OFF	OFF	OFF
NOTES.					

NOTES:

E8: Determines the signal on pin 5 of the Fieldbus 9-pin D-Sub Connector. The position of the jumper depends on the COMX module installed/option ordered.

E10: Adds 120 Ω termination resistor for DeviceNet communication lines

E11: Adds 120 Ω termination resistor for CANopen communication lines

E12: Adds 110 Ω termination resistor for CC-Link communication lines

Connector Pinouts

Fieldbus Port (J4)

Protocol Pin No.	PROFIBUS	DeviceNet	CANopen	CC-Link
1		+24 V Power Supply		CC-Link, Shield
2	Positive power supply	CAN High-Signal	CAN_L Bus Line	CC-Link, Function Ground
3	Receive / Send Data-P	Reference potential	CAN Ground	CC-Link, Data A
4	Control			
5	Reference potential	Shield		CC-Link, Data Ground
6	Positive power supply	CAN High-Signal	CAN_L Bus Line	CC-Link, Function Ground
7			CAN_H Bus Line	
8	Receive / Send Data-N			
9		CAN Low-Signal		CC-Link, Data B
NOTES	E8, Jumpered 1-2	E8, 2-3 Jumpered E10 Jumpered	E11 Jumpered	E8, Jumpered 2-3 E12 Jumpred

Real-time Ethernet Ports (Ethernet 0 & Ethernet 1)

Pin No.	Symbol	Description
1	RX+	Receive+
2	RX-	Receive-
3	TX+	Transmit+
4		
5		
6	TX-	Transmit–
7		
8		

Diagnostics Port (Micro A USB)

Pin No.	Symbol	Description								
1	VBUS	+5 VDC (Not connected to ACC-72EX +5 VDC)								
2	D-	Data -								
3	D+	Data +								
4	GND	Ground Reference (Connected to ACC-72EX and UMAC's Digital Ground)								

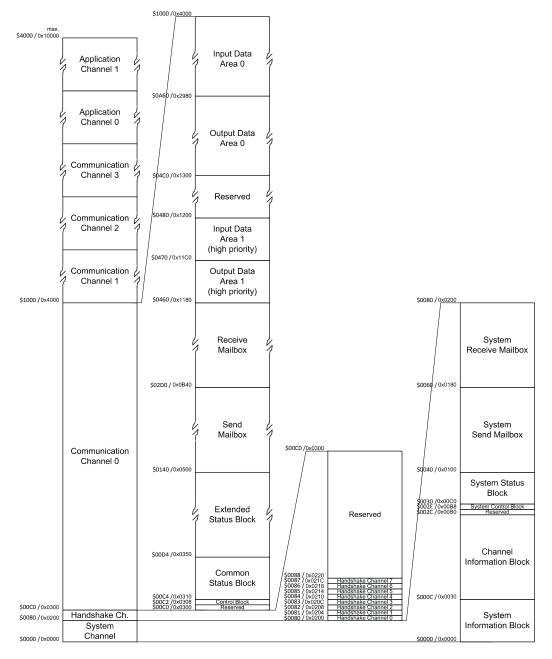
Note:

The USB connection is not a galvanically isolated connection. The ground of the PC will be connected to the ground of the UMAC system through the USB connection, which can damage components on the PC and/or ACC-72EX. Make sure that there is no potential difference between the grounds on both ends.

DPRAM MEMORY MAP

Below is the standard memory map of address offsets found in the DPRAM of the ACC-72EX module. Start and end addresses for each of the memory blocks are specified both in Hilscher offset (0x notation for hexadecimal) and Turbo PMAC offset (\$ notation for hexadecimal) notation. The memory map shown here is the standard memory map. Different COMX modules may have different memory maps. Please refer to the Hilscher manual for each COMX module for detailed information.

These registers should be read from and written to using M-Variables which point to the lower 16 bits of the X/Y-memory with an offset from the base address that is configured with E3. Handshake and System registers are common between all protocols, and others can be auto-generated using the "ACC-72EX Setup Assistant" software.



DPRAM Blocks

In the Hilscher COMX module DPRAM, the system channel and the handshake channel are always present. These channels are used for communicating with the firmware on the COMX module, from this point on referred to as "netX."

The system channel provides information about the state of the COMX module operating system, netX, and the structure of the dual-port memory. It allows basic communication via system mailboxes.

The handshake channel provides a bit toggle mechanism that insures synchronizing data transfer between the UMAC and COMX module. All handshake cells from system, communication, and application channels are brought together in this one location.

Next are the communication and application channels. A communication channel provides network access and occupies an area of the COMX dual-port memory with process, non-cyclic, and diagnostic data. An application channel can be used for any functionality that may be executed in the context of the netX operating system. The application channels are not supported by COMX modules at the time of writing this manual.

DPRAM Suggested Macro Names

ACC-72EX Setup Assistant software, available through the Tools menu in PEWIN32PRO2 software, provides a complete overview of blocks and sub-blocks available to each ACC-72EX. Under each section is a list of macro names provided for M-Variable definition.

For more information on structures and data registers in COMX modules, please refer to references introduced in appendix A of this manual.

System Channel

The System Channel is the first of the channels in the dual-port memory and starts at address offset \$0000. It holds information about the system itself (netX, netX operating system) and provides a mailbox transfer mechanism for system-related messages or packets.

ACC-72EX Setup Assistant software uses the data available in this channel to generate the information in the memory map output file.

System Information Block

The first block of information allows identification of the netX dual memory; it is used for testing proper communication. The first 4 registers hold character values for "netX" (110, 101, 116, 88). If these values are reading properly, the DPRAM communication is in working condition.

	Hilscher Documentation	ACC-72EX Setup Assistant						
	abCookie[4]	SI_abCookie_0 SI_abCookie_3_						
	ulDpmTotalSize	SI_ulDpmTotalSize						
	ulDeviceNumber	SI_ulDeviceNumber						
×	ulSerialNumber	SI_ulSerialNumber						
300	ausHwOptions[4]	SI_ausHwOptions_0 SI_ausHwOptions_3_						
on E	usManufacturer	SI_usManufacturer						
System Information Block	usProductionDate	SI_usProductionDate						
L L L	ulLicenseFlags1	SI_ulLicenseFlags1						
nfo	ulLicenseFlags2	SI_ulLicenseFlags2						
Ē	usNetxLicenseID	SI_usNetxLicenseID						
ste	usNetxLicenseFlags	SI_usNetxLicenseFlags						
Ś	usDeviceClass	SI_usDeviceClass						
	bHwRevision	SI_bHwRevision						
	bHwCompatibility	SI_bHwCompatibility						
	bDevIdNumber	SI_bDevIdNumber						

Channel Information Block

The system block includes information about all the other channels and their availability on the COMX module. This information is used to locate and identify different channels in the system.

	Hilscher Documentation	ACC-72EX Setup Assistant					
e	bChannelType	SCI_bChannelType					
uu	bSizePositionOfHandshake	SCI_bSizePositionOfHandshake					
Cha	bNumberOfBlocks	SCI_bNumberOfBlocks					
/stem (Inform	ulSizeOfChannel	SCI_ulSizeOfChannel					
	usSizeOfMailbox	SCI_usSizeOfMailbox					
sy	usMailboxStartOffset	SCI_usMailboxStartOffset					

	Hilscher Documentation	ACC-72EX Setup Assistant					
a i	bChannelType	HCI_bChannelType					
Handshake Channel Info	ulSizeOfChannel	HCI_ulSizeOfChannel					

	Hilscher Documentation	ACC-72EX Setup Assistant					
lər	bChannelType	CCxI_bChannelType					
iani	bChannelId	CCxI_bChannelId					
on Ch	bSizePositionOfHandshake	CCxI_bSizePositionOfHandshake					
ation Ch mation	bNumberOfBlocks	CCxI_bNumberOfBlocks					
icati	ulSizeOfChannel	CCxl_ulSizeOfChannel					
Infor	usCommunicationClass	CCxl_usCommunicationClass					
шu	usProtocolClass	CCxI_usProtocolClass					
Ĉ	usConformanceClass	CCxl_usConformanceClass					
Note: x in	ote: x in MACRO name is replaced by Application Channel number 0 3						

	Hilscher Documentation	ACC-72EX Setup Assistant					
_ ·	bChannelType	ACxI_bChannelType					
ication nel Info	bChannelId	ACxI_bChannelId					
	bSizePositionOfHandshake	ACxI_bSizePositionOfHandshake					
Appl Chanr	bNumberOfBlocks	ACxI_bNumberOfBlocks					
Cr⊳	ulSizeOfChannel	ACxI_ulSizeOfChannel					
Note: x in	Note: x in MACRO name is replaced by Application Channel number 0 1						

System Control Block

The system control block is used by UMAC to force netX to execute certain commands in the future. Currently, there are no such commands defined.

	Hilscher Documentation	ACC-72EX Setup Assistant						
System Control Block	ulSystemCommandCOS	SCtrl_ulSystemCommandCOS						

System Status Block

The system status block provides information about the staus of the netX firmware.

	Hilscher Documentation	ACC-72EX Setup Assistant					
к	ulSystemCOS	SStat_ulSystemCOS					
Block	ulSystemStatus	SStat_ulSystemStatus					
SL	ulSystemError	SStat_ulSystemError					
Statu	ulBootError	SStat_ulBootError					
٦	ulTimeSinceStart	SStat_ulTimeSinceStart					
Syster	usCpuLoad	SStat_usCpuLoad					
sγ	ulHWFeatures	SStat_ulHWFeatures					

System Mailbox

The system mailbox is the "window" to the operating system. It is always present even if no firmware is loaded. For more information about using system send/receive mailboxes, please see the examples shown in the following chapters. A complete list of functions, which can be accessed using the mailboxes, can be found in the netX Dual-Ported Memory Interface document available from Hilscher.

	Hilscher Documentation	ACC-72EX Setup Assistant					
	usPackagesAccepted	SSMB_usPackagesAccepted					
Mailbox	ulDest	SSMB_ulDest					
	ulSrc	SSMB_ulSrc					
Лаі	ulDestId	SSMB_ulDestId					
∠ p	ulSrcId	SSMB_ulSrcId					
Send	ulLen	SSMB_ulLen					
ъ К	ulld	SSMB_ulld					
Block	ulState	SSMB_ulState					
System	ulCmd	SSMB_ulCmd					
yst	ulExt	SSMB_ulExt					
S	ulRout	SSMB_ulRout					
		SSMB_ultData0 SSMB_ultData20					

	Hilscher Documentation	ACC-72EX Setup Assistant						
	usWaitingPackages	SRMB_usWaitingPackages						
ŏ	ulDest	SRMB_ulDest						
Receive Mailbox	ulSrc	SRMB_ulSrc						
ž	ulDestId	SRMB_ulDestId						
eive	ulSrcId	SRMB_ulSrcId						
ece	ulLen	SRMB_ulLen						
	ulld	SRMB_ulld						
locl	ulState	SRMB_ulState						
лB	ulCmd	SRMB_ulCmd						
System Block	ulExt	SRMB_ulExt						
Sys	ulRout	SRMB_ulRout						
		SRMB_ultData0 SRMB_ultData20						

Handshake Channel

The handshake channel provides a mechanism that allows the synchronizing of data transfer between the UMAC CPU and ACC-72EX dual-port memory. The handshake channel brings all handshake registers from other channel blocks together in one location. The handshake register could be moved from the handshake block to the beginning of each of the communication channels.

There are three types of handshake cells, described below.

System Handshake Cells

System handshake flags are used to synchronize data transfer between the ACC-72EX Hilscher Module and UMAC via the system mailbox and to handle certain changes of state function. They also hold information about the status of the ACC-72EX Hilscher module and can be used to execute certain commands in the module (for a module-wide reset, for example).

There are two sets of system flags. One set is dedicated for netX writes and is read by UMAC, and the other one is designated for UMAC writes. netX is continuously reading the second set.

netX System Flags

The ACC-72EX Hilscher module firmware writes to the netX system register; UMAC reads this register. The netX system register is located at address offset \$80 in the dual-port memory.

bNe	NetxFlags – netX writes, UMAC reads															
Bit	15 14 13 12 11 10 9 8 7 6										5	4	3	2	1	0
Function					Rese	rved					NSF_RECV_MBX_CMD	NSF_SEND_MBX_ACK	NSF_NETX_COS_CMD	NSF_HOST_COS_ACK	NSF_ERROR	NSF_READY

netX System Flags **bNetxFlags** (ACC-72EX ⇔ UMAC)

Bit No.	Definition / Description
	Ready (NSF_READY)
0	The Ready flag is set as soon as the COMX has initialized itself properly and passed its self-
	test. When the flag is set, the netX is ready to accept packets via the system mailbox. If
	cleared, the netX does not accept any packages.
	Error (NSF_ERROR)
	The Error flag is set when the netX has detected an internal error condition. This is
1	considered to be a fatal error. The Ready flag is cleared and the netX operating system is
	stopped. An error code helping to identify the issue is placed in the ulSystemError variable
	in the system status block.
	Host Change Of State Acknowledge (NSF_HOST_COS_ACK)
2	The Host Change of State Acknowledge flag is set when the netX acknowledges a command
_	from the host system. This flag is used together with the Host Change of State Command
	flag in the host system flags.
	netX Change Of State Command (NSF_NETX_COS_CMD)
3	The netX Change of State Command flag is set if the netX signals a change of its state to the
C	host system. Details of what has changed can be found in the ulSystemCOS variable in the
	system control block.
	Send Mailbox Acknowledge (NSF_SEND_MBX_ACK)
4	Both the Send Mailbox Acknowledge flag and the Send Mailbox Command flag are used
	together to transfer non-cyclic packages between the UMAC and the netX.
	Receive Mailbox Command (NSF_RECV_MBX_CMD)
5	Both the Receive Mailbox Command flag and the Receive Mailbox Acknowledge flag are
	used together to transfer non-cyclic packages between the netX and UMAC.
6, 7 15	6, 7 15 Reserved, set to zero

	Hilscher Documentation	ACC-72EX Setup Assistant					
s je	bNetxFlags	HCSC_bNetxFlags					
hake lags	NSF_READY	HCSC_NSF_READY					
ndshake ter m Flags	NSF_ERROR	HCSC_NSF_ERROR					
m Hands Register System l	NSF_HOST_COS_ACK	HCSC_NSF_HOST_COS_ACK					
Re	NSF_NETX_COS_CMD	HCSC_NSF_NETX_COS_CMD					
Syste netX	NSF_SEND_MBX_ACK	HCSC_NSF_SEND_MBX_ACK					
s u	NSF_RECV_MBX_CMD	HCSC_NSF_RECV_MBX_CMD					

Host System Flags

The host system flags are written by UMAC; the netX reads these flags. The host system register is located at address offset \$81 in the dual-port memory.

bH	ostFlag	stFlags – UMAC writes, netX reads														
Bit	15	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0											0			
Function					Rese	rved					HSF_RECV_MBX_ACK	HSF_SEND_MBX_CMD	HSF_NETX_COS_ACK	HSF_HOST_COS_CMD	HSF_BOOTSTART	HSF_RESET

Host System Flags **bHostFlags** (UMAC ⇒ ACC-72EX)

Bit No.	Definition / Description
	Reset (HSF_RESET)
0	The Reset flag is set by the UMAC to execute a system wide reset. This forces the system to
	restart. All network connections are interrupted immediately regardless of their current state.
	Bootstart (HSF_BOOTSTART)
1	If set during reset, the Boot-Start flag forces the netX to stay in boot loader mode; a firmware
1	that may reside in the context of the operating system rcX is not started. If cleared during
	reset, the operating system will start the firmware if available.
	Host Change Of State Command (HSF_HOST_COS_CMD)
2	The Host Change of State Command flag is set by the UMAC to signal a change of its state
2	to the netX. Details of what has changed can be found in the ulSystemCommandCOS
	variable in the system control block.
	netX Change Of State Acknowledge (HSF_NETX_COS_ACK)
3	The netX Change of State Acknowledge flag is set by the UMAC to acknowledge the new
5	state of the netX. This flag is used together with the netX Change of State Command flag in
	the netX system flags.
	Send Mailbox Command (HSF_SEND_MBX_CMD)
4	Both the Send Mailbox Command flag and the Send Mailbox Acknowledge flag are used
	together to transfer non-cyclic packages between the UMAC and the netX.
	Receive Mailbox Acknowledge (HSF_RECV_MBX_ACK)
5	Both the Receive Mailbox Acknowledge flag and the Receive Mailbox Command flag are
	used together to transfer non-cyclic packages between the netX and the UMAC.
6, 7	6, 7 15 Reserved; set to zero
15	

	Hilscher Documentation	ACC-72EX Setup Assistant					
em em	bHostFlags	HCSC_bHostFlags					
dshake Syster	HSF_RESET	HCSC_HSF_RESET					
nds st Sy	HSF_BOOTSTART	HCSC_HSF_BOOTSTART					
Hand Host Flags	HSF_HOST_COS_CMD	HCSC_HSF_HOST_COS_CMD					
5 5	HSF_NETX_COS_ACK	HCSC_HSF_NETX_COS_ACK					
Syster Registe	HSF_SEND_MBX_CMD	HCSC_HSF_SEND_MBX_CMD					
S, Re	HSF_RECV_MBX_ACK	HCSC_HSF_RECV_MBX_ACK					

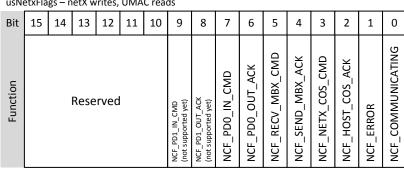
Communication Channel Handshake Cells

The channel handshake register is used to indicate the status of the protocol stack as well as execute certain commands in the protocol stack (e.g. reset a channel or synchronization of process data). The mailbox flags are used to send and receive non-cyclic messages via the channel mailboxes.

There are two sets of Communication Channel flags. One set is dedicated for netX writes; UMAC continually reads this. The other set is designated for UMAC writes; netX continuously reads this.

netX Communication Flags

This flag register is organized as a bit field. The netX protocol stack writes to the register to control data synchronization via the mailbox system and the process data image. It also informs the UMAC about its current network state. The UMAC reads this register.



usNetxFlags - netX writes, UMAC reads

Communication Channel Flags usNetXFlags (ACC-72EX ⇒ UMAC)

Bit	Definition / Description
No.	Å
	Communicating (NCF_COMMUNICATING)
	The NCF_COMMUNICATING flag is set if the protocol stack has successfully opened a
0	connection to at least one of the configured network slaves (for master protocol stacks),
0	respectively has an open connection to the network master (for slave protocol stacks). If cleared,
	the input data should not be evaluated, because it may be invalid, old or both. At
	initialization time, this flag is cleared.
	Error (NCF_ERROR)
	The NCF_ERROR flag signals an error condition that is reported by the protocol stack. It could
1	indicate a network communication issue or something to that effect. The corresponding error
	code is placed in the ulCommunicationError variable in the common status block. At
	initialization time, this flag is cleared.
	Host Change Of State Acknowledge (NCF_HOST_COS_ACK)
2	The NCF_HOST_COS_ACK flag is used by the protocol stack indicating that the new state of
	the UMAC has been read. At initialization time, this flag is cleared.
	netX Change Of State Command (NCF_NETX_COS_CMD)
	The NCF_NETX_COS_CMD flag signals a change in the state of the protocol stack. The new
2	state can be found in the ulCommunicationCOS register in the common status block. In return
3	the UMAC program then toggles the HCF_NETX_COS_ACK flag in the host communication
	flags acknowledging that the new protocol state has been read. At initialization time, this flag is
	cleared.
	Send Mailbox Acknowledge (NCF_SEND_MBX_ACK)
4	Both the NCF_SEND_MBX_ACK flag and the HCF_SEND_MBX_CMD flag are used
4	together to transfer non-cyclic packages between the protocol stack and the UMAC programs.
	At initialization time, this flag is cleared.
	Receive Mailbox Command (NCF_RECV_MBX_CMD)
5	Both the NCF_RECV_MBX_CMD flag and the HCF_RECV_MBX_ACK flag are used
5	together to transfer non-cyclic packages between the UMAC programs and the protocol stack.
	At initialization time, this flag is cleared.
	Process Data 0 Out Acknowledge (NCF_PD0_OUT_ACK)
6	Both the NCF_PD0_OUT_ACK flag and the HCF_PD0_OUT_CMD flag are used together to
0	transfer cyclic output data from the UMAC to the protocol stack. At initialization time, this flag
	may be set, depending on the data exchanged mode.
	Process Data 0 In Command (NCF_PD0_IN_CMD)
7	Both the NCF_PD0_IN_CMD flag and the HCF_PD0_IN_ACK flag are used together to
,	transfer cyclic input data from the protocol stack to the UMAC. At initialization time, this flag
	may be set, depending on the data exchanged mode.
	Process Data 1 Out Acknowledge (NCF_PD1_OUT_ACK, not supported yet)
8	Both the NCF_PD1_OUT_ACK flag and the HCF_PD1_OUT_CMD flag are used together to
0	transfer output cyclic data from the UMAC to the protocol stack. At initialization time, this flag
	may be set, depending on the data exchanged mode.
	Process Data 1 In Command (NCF_PD1_IN_CMD, not supported yet)
9	Both the NCF_PD1_IN_CMD flag and the HCF_PD1_IN_ACK flag are used together to
	transfer cyclic input data from the protocol stack to the UMAC. At initialization time, this flag
	may be set, depending on the data exchange mode.
1015	Reserved, set to 0

	Hilscher Documentation	ACC-72EX Setup Assistant							
	usNetxFlags	HCCCx_usNetxFlags							
gs	NCF_COMMUNICATING	HCCCx_NCF_COMMUNICATING							
Flags er	NCF_ERROR	HCCCx_NCF_ERROR							
cation Regist	NCF_HOST_COS_ACK	HCCCx_NCF_HOST_COS_ACK							
	NCF_NETX_COS_CMD	HCCCx_NCF_NETX_COS_CMD							
	NCF_SEND_MBX_ACK	HCCCx_NCF_SEND_MBX_ACK							
nm sha	NCF_RECV_MBX_CMD	HCCCx_NCF_RECV_MBX_CMD							
X Communi Handshake	NCF_PD0_OUT_ACK	HCCCx_NCF_PDx_OUT_ACK							
netX (Ha	NCF_PD0_IN_CMD	HCCCx_NCF_PDx_IN_CMD							
ne	NCF_PD1_OUT_ACK	HCCCx_NCF_PD1_OUT_ACK							
	NCF_PD1_IN_CMD	HCCCx_NCF_PD1_IN_CMD							
Note: x i	n MACRO name is replaced by Communication C	hannel number 03							

Host Communication Flags

This flag register is organized as a bit field. UMAC writes to this register to control data synchronization via the mailbox system and the process data image. The netX protocol stack reads this register.

usin	is a low rest reads															
Bit	t 15 14 13 12 11 10							8	7	6	5	4	3	2	1	0
Function			Rese	rved	I		HCF_PD1_IN_ACK (not supported yet)	HCF_PD1_OUT_CMD (not supported yet)	HCF_PD0_IN_ACK	HCF_PD0_OUT_CMD	HCF_RECV_MBX_ACK	HCF_SEND_MBX_CMD	HCF_NETX_COS_ACK	HCF_HOST_COS_CMD		

usHostFlags – UMAC writes, netX reads

Communication Channel Flags usHostFlags (UMAC ⇒ ACC-72EX)

Bit No.	Definition / Description
0, 1	Reserved, set to 0
2	Host Change Of State Command (HCF_HOST_COS_CMD) The HCF_HOST_COS_CMD flag signals a change in the state of the UMAC. A new state is set in the ulApplicationCOS variable in the communication control block. The protocol stack on the netX then toggles the NCF_HOST_COS_ACK flag in the netX communication flags back acknowledging that the new state has been read. At initialization time, this flag is cleared.
3	Host Change Of State Acknowledge (HCF_NETX_COS_ACK) The HCF_NETX_COS_ACK flag is used by UMAC to indicate that the new state of the protocol stack has been read. At initialization time, this flag is cleared.
4	Send Mailbox Command (HCF_SEND_MBX_CMD) Both the HCF_SEND_MBX_CMD flag and the NCF_SEND_MBX_ACK flag are used together to transfer non-cyclic packages between the UMAC and the protocol stack. At initialization time, this flag is cleared.
5	Receive Mailbox Acknowledge (HCF_RECV_MBX_ACK) Both the HCF_RECV_MBX_ACK flag and the NCF_RECV_MBX_CMD flag are used together to transfer non-cyclic packages between the protocol stack and the UMAC. At initialization time, this flag is cleared.
6	Process Data 0 Out Command (HCF_PD0_OUT_CMD) Both the HCF_PD0_OUT_CMD flag and the NCF_PD0_OUT_ACK flag are used together to transfer cyclic output data from the UMAC to the protocol stack. At initialization time, this flag may be set, depending on the data exchanged mode.
7	Process Data 0 In Acknowledge (HCF_PD0_IN_ACK) Both the HCF_PD0_IN_ACK flag and the NCF_PD0_IN_CMD flag are used together to transfer cyclic input data from the protocol stack to the UMAC. At initialization time, this flag may be set, depending on the data exchanged mode.
8	Process Data 1 Out Command (HCF_PD1_OUT_CMD, not supported yet) Both the HCF_PD1_OUT_CMD flag and the NCF_PD1_OUT_ACK flag are used together to transfer cyclic output data from the UMAC to the protocol stack. At initialization time, this flag may be set, depending on the data exchanged mode.
9	Process Data 1 In Acknowledge (HCF_PD1_IN_ACK, not supported yet) Both the HCF_PD1_IN_ACK flag and the NCF_PD1_IN_CMD flag are used together to transfer cyclic input data from the protocol stack to the UMAC. At initialization time, this flag may be set, depending on the data exchanged mode.
10 15	Reserved, set to 0

	Hilscher Documentation	ACC-72EX Setup Assistant							
gs	usHostFlags	HCCCx_usHostFlags							
Flags er	HCF_HOST_COS_CMD	HCCCx_HCF_HOST_COS_CMD							
cation Registe	HCF_NETX_COS_ACK	HCCCx_HCF_NETX_COS_ACK							
cation Regist	HCF_SEND_MBX_CMD	HCCCx_HCF_SEND_MBX_CMD							
Communi andshake	HCF_RECV_MBX_ACK	HCCCx_HCF_RECV_MBX_ACK							
t Communi Handshake	HCF_PD0_OUT_CMD	HCCCx_HCF_PDx_OUT_CMD							
Cor	HCF_PD0_IN_ACK	HCCCx_HCF_PDx_IN_ACK							
Host e	HCF_PD1_OUT_CMD	HCCCx_HCF_PD1_OUT_CMD							
Ĥ	HCF_PD1_IN_ACK	HCCCx_HCF_PD1_IN_ACK							
Note: x i	in MACRO name is replaced by Communication C	hannel number 0 3							

Application Handshake Cells

Although these cells are not supported yet, the following structure groups have been defined for backward compatibility as a placeholder:

netX Communication Flags

Host Communication Flags

Communication Channel

The communication channel structure is mainly dependent on the protocol firmware and COMX module. However, there are common sub-block structures which are common to all protocols.

Control Block

The control block of a dual-port memory features a Watchdog function to allow the operating system running on the netX to supervise the host application and vice versa. The control area is always present in dual-port memory. This block can also be read using the mailbox interface.

Application Change of State Register

The Application Change of State Register is a bit field. The UMAC uses this field to send commands to the communication channel. Changing flags in this register requires the UMAC to toggle the Host Change of State Command flag in the Host Communication Flags register, and then the netX protocol stack will recognize the change.

ulAp	pplicationCOS – UMAC writes, netX reads															
Bit	31 30 29 11 10 9								7	6	5	4	3	2	1	0
Function			Re	serv	ed			RCX_APP_COS_DMA_ENABLE	RCX_APP_COS_DMA	RCX_APP_COS_LOCK_CONFIG_ENABLE	RCX_APP_COS_LOCK_CONFIG	RCX_APP_COS_INIT_ENABLE	RCX_APP_COS_INIT	RCX_APP_COS_BUS_ON_ENABLE	RCX_APP_COS_BUS_ON	RCX_APP_COS_APP_READY

Bit	Definition / Description
No.	
0	Application Ready (RCX_APP_COS_APP_READY, not supported yet)
0	If set, the UMAC indicates to the protocol stack that its state is Ready.
	Bus On (RCX_APP_COS_BUS_ON)
	Using the Bus On flag, the UMAC allows or disallows the firmware to open network
1	connections. This flag is used with Bus On Enable flag below. If set, the netX firmware tries to
	open network connections; if cleared, no connections are allowed, and open connections are
	closed.
	Bus On Enable (RCX_APP_COS_BUS_ON_ENABLE)
2	The Bus On Enable flag is used together with the Bus On flag above. If set, this flag enables the
	execution of the Bus On command in the netX firmware.
	Initialization (RCX_APP_COS_INIT)
3	Setting the Initialization flag the UMAC forces the protocol stack to restart and evaluate the
5	configuration parameter again. All network connections are interrupted immediately regardless
	of their current state. If the database is locked, re-initializing the channel is not allowed.
	Initialization Enable (RCX_APP_COS_INIT_ENABLE)
4	The Initialization Enable flag is used together with the Initialization flag above. If set, this flag
	enables the execution of the Initialization command in the netX firmware.
	Lock Configuration (RCX_APP_COS_LOCK_CONFIG)
5	If set, UMAC does not allow the firmware to reconfigure the communication channel. The
5	database will be locked. The Configuration Locked flag in the channel status block shows if the
	current database has been locked.

Bit No.	Definition / Description
	Lock Configuration Enable (RCX_APP_COS_LOCK_CONFIG_ENABLE)
6	The Lock Configuration Enable flag is used together with the Lock Configuration flag above. If
	set, this flag enables the execution of the Lock Configuration command in the netX firmware.
	Turn on DMA Mode (RCX_APP_COS_DMA)
7	The UMAC sets this flag in order to turn on the DMA mode for the cyclic process data input /
	output image 0 (abPd0Output and abPd0Input).
	Turn on DMA Mode Enable (RCX_APP_COS_DMA_ENABLE)
8	The DMA Enable flag is used together with the DMA flag above. If set, this flag enables the
	execution of the DMA command in the netX firmware.
9 31	Reserved, set to 0

Device Watchdog Register

The protocol stack supervises the UMAC using a Watchdog function. If the UMAC fails to copy the value from the host Watchdog location to the device Watchdog location, the protocol stack assumes that the UMAC system has a problem and interrupts all network connections immediately, regardless of their current state.

	Hilscher Documentation	ACC-72EX Setup Assistant			
сk	RCX_APP_COS_APP_READY	CCx_RCX_APP_COS_APP_READY			
Block	RCX_APP_COS_BUS_ON	CCx_RCX_APP_COS_BUS_ON			
lo	RCX_APP_COS_BUS_ON_ENABLE	CCx_RCX_APP_COS_BUS_ON_ENABLE			
onti	RCX_APP_COS_INIT	CCx_RCX_APP_COS_INIT			
Ŭ	RCX_APP_COS_INIT_ENABLE	CCx_RCX_APP_COS_INIT_ENABLE			
tior	RCX_APP_COS_LOCK_CFG	CCx_RCX_APP_COS_LOCK_CFG			
iica	RCX_APP_COS_LOCK_CFG_ENA	CCx_RCX_APP_COS_LOCK_CFG_ENA			
Communication Control	RCX_APP_COS_DMA	CCx_RCX_APP_COS_DMA			
	RCX_APP_COS_DMA_ENABLE	CCx_RCX_APP_COS_DMA_ENABLE			
Ö	ulDeviceWatchdog	CCx_ulDeviceWatchdog			
Note: x i	Note: x in MACRO name is replaced by Application Channel number 0 3				

Common Status Block

The common status block contains information fields that are common to all protocol stacks. The status block is always present in dual-port memory. This block can also be read using the mailbox interface.

Communication Change of State Register

The Communication Change of State register is a bit field. It contains information about the current operating status of the communication channel and its firmware. Every time the status changes, the netX protocol stack toggles the netX Change of State Command flag in the netX communication flags register. The UMAC then has to toggle the netX Change of State Acknowledge flag back, acknowledging the new state.

ulCo	ulCommunicationCOS - netX writes, UMAC reads																
Bit	31	30	29		11	10	9	8	7	6	5	4	3	2	1	0	
Function				Rese	rved	I			RCX_COMM_COS_DMA	RCX_COMM_COS_RESTART_REQUIRED_ENABLE	RCX_COMM_COS_RESTART_REQUIRED	RCX_COMM_COS_CONFIG_NEW	RCX_COMM_COS_CONFIG_LOCKED	RCX_COMM_COS_BUS_ON	RCX_COMM_COS_RUN	RCX_COMM_COS_READY	

Bit No.	Definition / Description
	Ready (RCX_COMM_COS_READY)
0	The Ready flag is set as soon as the protocol stack is started properly. Then, the protocol stack
	awaits a configuration. As soon as the protocol stack is configured properly, the Running flag
	is set.
	Running (RCX_COMM_COS_RUN)
1	The Running flag is set when the protocol stack has been configured properly. Then the
1	protocol stack awaits a network connection. Now, both the Ready flag and the Running flag
	are set.
	Bus On (RCX_COMM_COS_BUS_ON)
	The Bus On flag is set to indicate to the UMAC whether or not the protocol stack has the
2	permission to open network connections. If set, the protocol stack has the permission to
	communicate on the network; if cleared, the permission was denied and the protocol stack will
	not open network connections.
	Configuration Locked (RCX_COMM_COS_CONFIG_LOCKED)
	The Configuration Locked flag is set if the communication channel firmware has locked the
3	configuration database against being overwritten. Reinitializing the channel is not allowed in
	this state. To unlock the database, the application has to clear the Lock Configuration flag in
	the control block.
	Configuration New (RCX_COMM_COS_CONFIG_NEW)
4	The Configuration New flag is set by the protocol stack to indicate that a new configuration
	became available, but has not yet been activated. This flag may be set together with the
	Restart Required flag.
	Restart Required (RCX_COMM_COS_RESTART_REQUIRED)
	The Restart Required flag is set when the channel firmware requests to be restarted. This flag
5	is used together with the Restart Required Enable flag below. Restarting the channel firmware
	may become necessary if a new configuration was downloaded from the UMAC or if a
	configuration upload via the network took place.
-	Restart Required Enable (RCX_COMM_COS_RESTART_REQUIRED_ENABLE)
6	The Restart Required Enable flag is used together with the Restart Required flag above. If set,
	this flag enables the execution of the Restart Required command in the netX firmware.
_	DMA Mode On (RCX_COMM_COS_DMA)
7	The protocol stack sets this flag in order to signal to the UMAC that the DMA mode is turned
	on.
8 31	Reserved, set to 0

Communication State

The communication state field contains current device network communication status information. Depending on the implementation, all or a subset of the definitions below is supported:

Value	Definition / Description
\$0	UNKNOWN
\$1	OFFLINE
\$2	STOP
\$3	IDLE
\$4	OPERATE

Communication Channel Error

This field holds the current error code of the communication channel. If the cause of error is resolved, the communication error field is set to zero (= RCX_S_OK) again. Not all of the error codes are supported in every implementation.

Watchdog Timeout

This field holds the configured Watchdog timeout value in milliseconds. The UMAC may set its Watchdog trigger interval accordingly. If the UMAC fails to copy the value from the host Watchdog location to the device Watchdog location, the protocol stack will interrupt all network connections immediately, regardless of their current state.

Handshake Mode

The protocol stack supports different handshake mechanisms to synchronize process data exchange with the UMAC. Depending on the configured mode, this mechanism insures data consistency over the entire data image and helps synchronize the UMAC with the network. This register holds the configured handshake mode.

Value	Definition / Description			
\$0	For compatibility reasons, this value is identical to 0x04 - Buffered Host Controlled IO			
φU	Data Transfer			
\$2	Buffered Device-Controlled I/O Data Transfer			
\$3	Uncontrolled Mode			
\$4	Buffered Host-Controlled IO Data Transfer			

Host Watchdog

The protocol stack supervises the UMAC via the Watchdog function. If the UMAC fails to copy the value from the device Watchdog location to the host Watchdog location, the protocol stack assumes that the UMAC has a problem and shuts down all network connections.

Error Count (All Implementations)

This field holds the total number of errors detected since power-up or after a reset. The protocol stack counts all sorts of errors in this field regardless if they were network-related or caused internally. The counter is cleared after a power cycle, reset, or channel initialization.

Error Log Indicator (All Implementations)

Not supported yet; the error log indicator field holds the number of entries in the internal error log. The field is set to zero if all entries are read from the log.

Number of Input Process Data Handshake Errors TBD

Number of Output Process Data Handshake Errors TBD

Number of Synchronization Handshake Errors

This counter will be incremented if the device detects a "not handled synchronization indication." This field is not supported yet.

Synchronization Status

This field is reserved for future use.

Slave State

The Slave State field indicates whether or not the master is in cyclic data exchange to all configured slaves. If there is at least one slave missing or if the slave has a diagnostic request pending, the status

Value	Definition / Description					
\$0	UNDEFINED					
\$1	OK. No Fault.					
\$2	\$2 FAILED. At least one slave failed					
Other values an	Other values are reserved					

changes to FAILED. For protocols that support non-cyclic communication only, the slave state is set to OK as soon as a valid configuration is found.

Slave Error Log Indicator

Not supported yet: the error log indicator field holds the number of entries in the internal error log. The field is set to zero if all entries are read from the log.

Number of Configured Slaves

The firmware maintains a list of slaves with which the master has to open a connection. This list is derived from the configuration database created by SYCON.net. This field holds the number of configured slaves.

Number of Active Slaves

The firmware maintains a list of slaves to which the master exchanges process data. This field holds the number of active slaves. Ideally, the number of active slaves is equal to the number of configured slaves. For certain fieldbus systems, it could be possible that a slave is shown as activated, but still has a problem (i.e. a diagnostic issue).

Number of Faulted Slaves

The firmware maintains a list of slaves that are missing on the network, although they may be configured, or are reporting a diagnostic issue. As long as those indications are pending and not serviced, the field holds a nonzero value. If no more diagnostic information is pending, the field is set to zero again.

	Hilscher Documentation	ACC-72EX Setup Assistant					
	RCX_COMM_COS_READY	CCx_RCX_COMM_COS_READY					
	RCX_COMM_COS_RUN	CCx_RCX_COMM_COS_RUN					
	RCX_COMM_COS_BUS_ON	CCx_RCX_COMM_COS_BUS_ON					
	RCX_COMM_COS_CONFIG_LOCKED	CCx_RCX_COMM_COS_CONFIG_LOCKED					
	RCX_COMM_COS_CONFIG_NEW	CCx_RCX_COMM_COS_CONFIG_NEW					
	RCX_COMM_COS_RESTART_REQ	CCx_RCX_COMM_COS_RESTART_REQ					
	RCX_COMM_COS_RESTART_REQ_ENA	CCx_RCX_COMM_COS_RESTART_REQ_ENA					
	RCX_COMM_COS_DMA	CCx_RCX_COMM_COS_DMA					
	ulCommunicationState	CCx_ulCommunicationState					
	ulCommunicationError	CCx_ulCommunicationError					
	usVersion	CCx_usVersion					
ock	usWatchdogTime	CCx_usWatchdogTime					
Blo	bPDInHskMode	CCx_bPDInHskMode					
Common Status Block	bPDInSource	CCx_bPDInSource					
Sta	bPDOutHskMode	CCx_bPDOutHskMode					
non	bPDOutSource	CCx_bPDOutSource					
μu	ulHostWatchdog	CCx_ulHostWatchdog					
CO	ulErrorCount	CCx_ulErrorCount					
	bErrorLogInd	CCx_bErrorLogInd					
	bErrorPDInCnt	CCx_bErrorPDInCnt					
	bErrorPDOutCnt	CCx_bErrorPDOutCnt					
	bErrorSyncCnt	CCx_bErrorSyncCnt					
	bSyncHskMode	CCx_bSyncHskMode					
	bSyncSource	CCx_bSyncSource					
	ulSlaveState	CCx_ulSlaveState					
	ulSlaveErrLogInd	CCx_ulSlaveErrLogInd					
	ulNumOfConfigSlaves	CCx_ulNumOfConfigSlaves					
	ulNumOfActiveSlaves	CCx_ulNumOfActiveSlaves					
	ulNumOfDiagSlaves	CCx_ulNumOfDiagSlaves					
Note: x	Note: x in MACRO name is replaced by Application Channel number 0 3						

Application Channel

The application channel is reserved for user specific implementations. An application channel is not yet supported.

Auto-Generated Dual-Ported Memory Map

ACC-72EX Setup Assistant Software, designed for use with Turbo PMAC, provides some level of automation in the identification of Hilscher COMX modules by generating a memory map file, suggested M-Variable definitions for important registers, and appropriate macro names.

Nemory Map Generator		Address Converter	
Connect to PM/	AC	ACC-72EX Base Address	\$6C000 •
		Hilscher Address Offset	0x0
Address Selection	\$6C000 •	Hilscher Data Width	32 👻
Starting M-Variable Number	6000 🜩	Hilscher Data Start Bit	0 💌
			Convert
Generate M-Variable D	efinitions	Equivalent PMAC Address	D:\$6C000
o PMAC is selected for communic onnection to PMAC was successf (ACC-72EX cards detected.			

Address Converter

The Address Converter section of the software allows conversion of offset, bit, and width parameters to PMAC memory addresses based on Hilscher documentation.

Memory Map Generator

The Memory Map Generator section of the software identifies the ACC-72EX cards in a UMAC system and generates both a memory map as a text file and M-Variable definition file with proper addressing, both of which indicate the ACC-72EX-based address selection.

Reading the Memory Map Text File

The output file from the software is a text file which can be read with any text editor software. This file includes generic information about the card.

Below is an example output file. Please see the notes in the right column for more information on specific items.

HilscherMemoryMap_\$6C000.txt File Co	ontent	Notes
Delta Tau Data Systems, Inc		
ACC-72EX Setup Assistant Au ACC-72EX Address: \$6C000	to-generated Memory Map	Base address of the ACC- 72EX selected in the Memory Map Generator section
netX Identification:	netX	The identification cookie provided by the netX firmware
Dual-Port Memory Size:	65536 bytes	111
Device Number:	1532100	
Serial Number:	21456	
Hardware Assembly Options:		
Port 0: Port 1: Port 2:	ETHERNET (internal Phy)	
Port 1:	ETHERNET (internal Phy) ETHERNET (internal Phy) NOT CONNECTED	
Port 2:	NOT CONNECTED	
Port 3:		
Hilscher Module Production		
	ormation: (PROFIBUS Master) (CANopen	
	(AS-Interface Master) (PROFINET IO RT	
	r) (EtherNet/IP Scanner) (SERCOS III	
Master) 1 Master License		
Tool License Information:		
Device Class:	COMX 100	
+ Block 0:		Block information
Channel Type:	System	For all blocks
Size of Channel: Channel Start Address:	512 bytes \$6C000	
	Cells: IN HANDSHAKE CHANNEL	
netX System Flags Adre	ss: X:\$6080.0.8	Calculates where the
<pre>netX System Flags Adre. Host System Flags Adre. Size of Handshake Cell. Size of Mailbox:</pre>	ss: X:\$6C080,8,8	handshake registers are
Size of Handshake Cell	s: 8 BITS	Located
Size of Mailbox:	256 bytes	
Mailbox Start address:	\$6C040	
Number of Subblocks:	5	
	AMIIC	Lists all channels' Sub-
Subblock 0: COMMON ST. Size:	176 bytes	Blocks
	\$6C000	PTOCK2
	IN - OUT (Bi-Directional)	
	DPM (Dual-Port Memory)	
	UNCONTROLLED	
Handshake Bit:	0	
Subblock 1: CONTROL		
Size:	8 bytes	
Start Offset:		
	OUT (Host System to netX)	
Transfer Type:	DPM (Dual-Port Memory)	
Handshake Mode:	UNCONTROLLED	
Handshake Bit:	0	
 Subblock 2: COMMON ST.	ATUS	
1 SUBBLOCK 2. COMMON ST		

Size: 64 bytes Start Offset: \$6C030 Transfer Direction: IN (netX to Host System) Transfer Type: DPM (Dual-Port Memory) Handshake Mode: UNCONTROLLED Handshake Bit: 0 |--- Subblock 3: MAILBOX Size: 128 bytes Start Offset: \$6C040 Transfer Direction: OUT (Host System to netX) Transfer Type: DPM (Dual-Port Memory) Handshake Mode: BUFFERED, HOST CONTROLLED Handshake Bit: 4 |--- Subblock 4: MAILBOX 128 bytes Size: Start Offset: \$6C060 Transfer Direction: IN (netX to Host System) DPM (Dual-Port Memory) Transfer Type: Handshake Mode: UNKNOWN Handshake Bit: 5 + Block 1: Channel Type: Handshake Т Size of Channel: 256 bytes Т Channel Start Address: \$6C080 Т + Block 2: Channel Type: Communication 15616 bytes \$6C0C0 Size of Channel: Channel Start Address: Position of Handshake Cells: IN HANDSHAKE CHANNEL Size of Handshake Cells: 16 BITS NetX Handshake Register: Y:\$6C082,0,16 NetX Handshake Register: Host Handshake Register: X:\$6C082,0,16 Communication Class: SCANNER Communication Class: Protocol Class: IO-DEVICE Conformance Class: 0 Number of Subblocks: 9 |--- Subblock 0: CONTROL Size: 8 bytes Start Offset: \$6C0C2 Transfer Direction: OUT (Host System to netX) Transfer Type: DPM (Dual-Port Memory) Handshake Mode: UNCONTROLLED Handshake Bit: 0 -- Subblock 1: COMMON STATUS 64 bytes Size: Start Offset: \$6C0C4 Transfer Direction: IN (netX to Host System) Transfer Type: DPM (Dual-Port Memory) Handshake Mode: UNCONTROLLED Handshake Mode: Handshake Bit: 0 --- Subblock 2: EXTENDED STATUS 432 bytes Size: Start Offset: \$6C0D4 Transfer Direction: IN (netX to Host System) Transfer Type: DPM (Dual-Port Memory) Handshake Mode: UNCONTROLLED Handshake Bit: 0 - Subblock 3: MAILBOX 1600 bytes Size: Start Offset: \$6C140 Transfer Direction: OUT (Host System to netX)

Transfer Type: DPM (Dual-Port Memory) Handshake Mode: BUFFERED, HOST CONTROLLED Handshake Bit: 4 |--- Subblock 4: MAILBOX Size: 1600 bytes Start Offset: \$6C2D0 Transfer Direction: IN (netX to Host System) Transfer Type: DPM (Dual-Port Memory) Handshake Mode: UNKNOWN Handshake Bit: 5 |--- Subblock 5: PROCESS DATA IMAGE Size: 5760 bytes Start Offset: \$6C4C0 Transfer Direction: OUT (Host System to netX) Transfer Type: DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED Handshake Mode: Handshake Bit: 6 |--- Subblock 6: PROCESS DATA IMAGE 5760 bytes Size: Start Offset: \$6CA60 Transfer Direction: IN (netX to Host System) Transfer Type: DPM (Dual-Port Memory) Handshake Mode: BUFFERED, HOST CONTROLLED Handshake Bit: 7 |--- Subblock 7: HIGH PRIORITY DATA IMAGE Size: 64 bytes Start Offset: \$6C460 Transfer Direction: OUT (Host System to netX) Transfer Type: DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED Handshake Mode: Handshake Bit: 8 |--- Subblock 8: HIGH PRIORITY DATA IMAGE Size: 64 bytes Start Offset: \$6C470 Transfer Direction: IN (netX to Host System) Transfer Type: DPM (Dual-Port Memory) Handshake Mode: BUFFERED, HOST CONTROLLED Handshake Bit: 9 + Block 3: Channel Type: Communication Size of Channel: 15616 bytes Channel Start Address: \$6D000 Position of Handshake Cells: IN HANDSHAKE CHANNEL Size of Handshake Cells: 16 BITS NetX Handshake Register: Y:\$6C083,0,16 X:\$6C083,0,16 Host Handshake Register: Communication Class: MESSAGING Protocol Class: UNDEFINED Conformance Class: 0 Number of Subblocks: 9 -- Subblock 0: CONTROL Size: 8 bytes Start Offset: \$6D002 Transfer Direction: OUT (Host System to netX) Transfer Type: DPM (Dual-Port Memory) Handshake Mode: UNCONTROLLED Handshake Bit: 0 --- Subblock 1: COMMON STATUS 64 bytes Size: Start Offset: \$6D004 Transfer Direction: IN (netX to Host System) Transfer Type: DPM (Dual-Port Memory)

```
Handshake Mode:
                           UNCONTROLLED
      Handshake Bit:
                           0
--- Subblock 2: EXTENDED STATUS
                    432 bytes
      Size:
      Start Offset:
                          $6D014
      Transfer Direction: IN (netX to Host System)
      Transfer Type: DPM (Dual-Port Memory)
      Handshake Mode:
                          UNCONTROLLED
      Handshake Bit:
                          0
--- Subblock 3: MAILBOX
                          1600 bytes
      Size:
      Start Offset:
                           $6D080
      Transfer Direction: OUT (Host System to netX)
      Transfer Type: DPM (Dual-Port Memory)
      Handshake Mode:
                          BUFFERED, HOST CONTROLLED
      Handshake Bit:
                          4
--- Subblock 4: MAILBOX
      Size:
                          1600 bytes
      Start Offset:
                          $6D210
      Transfer Direction: IN (netX to Host System)
      Transfer Type:
                          DPM (Dual-Port Memory)
                          UNKNOWN
      Handshake Mode:
      Handshake Bit:
                          5
|--- Subblock 5: PROCESS DATA IMAGE
      Size:
                          5760 bytes
      Start Offset:
                          $6D400
      Transfer Direction: OUT (Host System to netX)
      Transfer Type:
                          DPM (Dual-Port Memory)
      Handshake Mode:
                          BUFFERED, HOST CONTROLLED
      Handshake Bit:
                          6
|--- Subblock 6: PROCESS DATA IMAGE
      Size:
                        5760 bytes
      Start Offset:
                           $6D9A0
      Transfer Direction: IN (netX to Host System)
      Transfer Type: DPM (Dual-Port Memory)
                          BUFFERED, HOST CONTROLLED
      Handshake Mode:
      Handshake Bit:
                          7
--- Subblock 7: HIGH PRIORITY DATA IMAGE
                          64 bytes
      Size:
      Start Offset:
                           $6D3A0
      Transfer Direction: OUT (Host System to netX)
      Transfer Type: DPM (Dual-Port Memory)
      Handshake Mode:
                          BUFFERED, HOST CONTROLLED
      Handshake Bit:
                          8
|--- Subblock 8: HIGH PRIORITY DATA IMAGE
      Size:
                          64 bytes
      Start Offset:
                          $6D3B0
      Transfer Direction: IN (netX to Host System)
      Transfer Type: DPM (Dual-Port Memory)
Handshake Mode: BUFFERED, HOST CONTROLLED
      Handshake Bit:
                          9
+ Block 4:
   Channel Type:
                                Undefined
   Size of Channel:
                                0 bytes
   Channel Start Address:
                                $6DF40
   Position of Handshake Cells: BEGINNING OF CHANNEL
   Size of Handshake Cells: NOT AVAILABLE
   NetX Handshake Register:
                                X:$6DF40
   Host Handshake Register:
                              X:$6DF40,8,0
                                UNDEFINED
   Communication Class:
    Protocol Class:
                                UNDEFINED
   Conformance Class:
                                0
```

	Number of Subblocks:	0	
+ B	lock 5:		
	Channel Type:	Undefined	
1	Size of Channel:	0 bytes	
1	Channel Start Address:	\$6DF40	
1	Position of Handshake Cells:	BEGINNING OF CHANNEL	
1	Size of Handshake Cells:	NOT AVAILABLE	
1	NetX Handshake Register:	X:\$6DF40	
1	Host Handshake Register:	X:\$6DF40,8,0	
1	Communication Class:	UNDEFINED	
1	Protocol Class:	UNDEFINED	
1	Conformance Class:	0	
1	Number of Subblocks:	0	

Suggested M-Variable Definition File Here is a sample macro name and suggested M-Variable definition file.

MacroNameDefinition_\$6C000.h File Content	
#define SI abCookie 0	M6000
#define SI_abCookie_1_	M6001
#define SI_abCookie_2_	M6002
#define SI abCookie 3	M6003
#define SI ulDpmTotalSize	M6004
#define SI_ulDeviceNumber	M6005
#define SI ulSerialNumber	M6006
#define SI ausHwOptions 0	M6007
#define SI ausHwOptions 1	M6008
#define SI_ausHwOptions_2_	M6009
#define SI ausHwOptions 3	M6010
#define SI usManufacturer	M6011
#define SI_usProductionDate	M6012
#define SI_ulLicenseFlags1	M6013
#define SI_ulLicenseFlags2	M6014
#define SI_usNetxLicenseID	M6015
#define SI_usNetxLicenseFlags	M6016
#define SI usDeviceClass	M6017
#define SI bHwRevision	M6018
#define SI bHwCompatibility	M6019
#define SI bDevIdNumber	M6020
#define SCI bChannelType	M6021
#define SCI bSizePositionOfHandshake	M6022
#define SCI bNumberOfBlocks	M6023
#define SCI_ulSizeOfChannel	M6024
#define SCI_usSizeOfMailbox	M6025
#define SCI_usMailboxStartOffset	M6026
#define HCI_bChannelType	M6027
#define HCI_ulSizeOfChannel	M6028
#define CC0I_bChannelType	M6029
#define CC0I_bChannelId	M6030
#define CCOI_bSizePositionOfHandshake	M6031
#define CC0I_bNumberOfBlocks	M6032
#define CC0I_ulSizeOfChannel	M6033
#define CCOI_usCommunicationClass	M6034
#define CCOI_usProtocolClass	M6035
#define CCOI_usConformanceClass	M6036
#define CC1I_bChannelType	M6037
#define CC1I_bChannelId	M6038
#define CC1I_bSizePositionOfHandshake	M6039
#define CC1I_bNumberOfBlocks	M6040
#define CC1I_ulSizeOfChannel	M6041
#define CC1I_usCommunicationClass	M6042
#define CC1I_usProtocolClass	M6043
#define CC1I_usConformanceClass	M6044
#define CC2I_bChannelType	M6045
#define CC2I_bChannelId	M6046
#define CC2I_bSizePositionOfHandshake	M6047
#define CC2I_bNumberOfBlocks	M6048

#define CC2I_ulSizeOfChannel	M6049
#define CC2I usCommunicationClass	M6050
	M6051
<pre>#define CC2I_usProtocolClass #define CC2I_usConformanceClass</pre>	
	M6052
#define CC3I bChannelType	M6053
#define CC3I bChannelId	M6054
#define CC3I bSizePositionOfHandshake	M6055
#define CC3I_bNumberOfBlocks	M6056
#define CC3I ulSizeOfChannel	M6057
#define CC3I usCommunicationClass	M6058
	M6059
<pre>#define CC3I_usProtocolClass #define CC3I_usConformanceClass</pre>	
	M6060
#define AC0I bChannelType	M6061
#define AC0I bChannelId	M6062
#define ACOI bSizePositionOfHandshake	M6063
#define AC0I_bNumberOfBlocks	M6064
#define ACOI_ulSizeOfChannel	M6065
#define AC1I bChannelType	M6066
#define AC11 bChannelId	M6067
<pre>#define AC1I_bSizePositionOfHandshake</pre>	M6068
#define AC1I_bNumberOfBlocks	M6069
#define AC11_ulSizeOfChannel	M6070
	M6071
<pre>#define SCtrl_ulSystemCommandCOS #define SStat_ulSystemCOS #define SStat_ulSystemStatus</pre>	
#deline Stat_uisystemCUS	M6072
#define SStat_ulSystemStatus	M6073
#define SStat_ulSystemError	M6074
#define SStat ulBootError	M6075
#define SStat_ulTimeSinceStart	M6076
#define SStat_usCpuLoad	M6077
#define SStat ulHWFeatures	M6078
#define SSMB usPackagesAccepted	M6079
#define SSMB_ulDest	M6080
#define SSMB_ulSrc	M6081
#define SSMB ulDestId	M6082
#define SSMB_ulSrcId	M6083
#define SSMB ullen	M6084
#define SSMB_ulId	M6085
#define SSMB ulState	M6086
#define SSMB_ulCmd	M6087
#define SSMB ulExt	M6088
#define SSMB_ulRout	M6089
#define SSMB ultData0	M6090
#define SSMB_ultData1	M6091
#define SSMB ultData2	M6092
=	
#define SSMB_ultData3	M6093
#define SSMB_ultData4	M6094
#define SSMB_ultData5	M6095
#define SSMB ultData6	M6096
#define SSMB_ultData7	M6097
#define SSMB_ultData8	M6098
#define SSMB_ultData9	M6099
#define SSMB ultData10	M6100
_	
#define SSMB_ultData11	M6101
#define SSMB_ultData12	M6102
#define SSMB ultData13	M6103
#define SSMB_ultData14	M6104
-	
#define SSMB_ultData15	M6105
#define SSMB_ultData16	M6106
#define SSMB ultData17	M6107
#define SSMB_ultData18	M6108
#define SSMB_ultData19	M6109
#define SSMB_ultData20	M6110
#define SRMB usWaitingPackages	M6111
#define SRMB_ulDest	M6112
#define SRMB ulSrc	M6113
_	
#define SRMB_ulDestId	M6114
#define SRMB ulSrcId	M6115
#define SRMB_ulLen	M6116
#define SRMB ulld	M6117
#define SRMB ulState	M6118
#define SRMB ulCmd	M6119

#define SRMe_ulsx: M6120 #define SRMe_ulsx: M6121 #define SRMe_ulsx: M6121 #define SRMe_ulsx: M6122 #define SRMe_ulsx: M6123 #define SRMe_ulsx: M6124 #define SRMe_ulsx: M6125 #define SRMe_ulsx: M6126 #define SRMe_ulsx: M6127 #define SRMe_ulsx: M6128		
Hedrim STMM_ultanata M1122 Vidrim STMM_ultanata M1123 Vidrim STMM_ultanata M1125 Vidrim STMM_ultanata M1126 Vidrim STMM_ultanata M1126 Vidrim STMM_ultanata M1126 Vidrim STMM_ultanata M1126 Vidrim STMM_ultanata M1128 Vidrim STMM_ultanata M1129 Vidrim STMM_ultanata M1121 Vidrim STMM_ultanata M1121 Vidrim STMM_ultanata M1121 Vidrim STMM_ultanata M1123 Vidrim STMM_ultanata <td>#define SRMB_ulExt</td> <td>M6120</td>	#define SRMB_ulExt	M6120
idefine SNM. ultatai Mi123 idefine SNM. ultatai Mi124 idefine SNM. ultatai Mi124 idefine SNM. ultatai Mi127 idefine SNM. ultatai Mi133 idefine SNM. ultatai Mi133 idefine SNM. ultatai Mi133 idefine SNM. ultatai Mi133 idefine SNM. ultatai Mi134 idefine SNM. ultatai Mi134 idefine SNM. ultatai Mi134 idefine SNM. ultatai Mi144 idefine Mi250 Mi144 Mi144	#define SRMB_ulRout	M6121
idefine SRM_ultDuta2 M0124 idefine SRM_ultDuta3 M0125 idefine SRM_ultDuta4 M0126 idefine SRM_ultDuta5 M0127 idefine SRM_ultDuta5 M0128 idefine SRM_ultDuta5 M0128 idefine SRM_ultDuta5 M0131 idefine SRM_ultDuta5 M0132 idefine SRM_ultDuta1 M0133 idefine SRM_ultDuta18 M0136 idefine SRM_ultDuta13 M0136 idefine SRM_ultDuta14 M0136 idefine SRM_ultDuta13 M0136 idefine SRM_ultDuta14 M0136 idefine SRM_ultDuta16 M0140 idefine SRM_ultDuta17 M0139 idefine SRM_ultDuta20 M0140 idefine SRM_ultDuta219 M0140 idefine SRM_ultDuta20 M0142 idefin	#define SRMB ultData0	M6122
idefine StMm_ultData3 M6125 idefine StMm_ultData5 M6126 idefine StMm_ultData5 M6127 idefine StMm_ultData6 M6128 idefine StMm_ultData7 M6128 idefine StMm_ultData7 M6128 idefine StMm_ultData7 M6128 idefine StMm_ultData10 M6133 idefine StMm_ultData12 M6136 idefine StMm_ultData14 M6136 idefine StMm_ultData14 M6137 idefine StMm_ultData14 M6138 idefine StMm_ultData15 M6137 idefine StMm_ultData14 M6138 idefine StMm_ultData17 M6140 idefine StMm_ultData18 M6137 idefine StMm_ultData17 M6142 idefine StMm_ultData17 M6142 idefine StMm_ultData17 M6143 idefine StMm_ultData18 M6142 idefine StMm_ultData17 M6144 idefine StMm_ultData18 M6144 idefine HCSC NST StMm NCAX M6144 idefine HCSC StMm NCAX M6146 idefine HCSC StMm NCAX M6146 <td>#define SRMB ultData1</td> <td>M6123</td>	#define SRMB ultData1	M6123
idefine SHM_ultData4 M6126 idefine SHM_ultData6 M6127 idefine SHM_ultData6 M6128 idefine SHM_ultData6 M6129 idefine SHM_ultData6 M6130 idefine SHM_ultData6 M6131 idefine SHM_ultData7 M6129 idefine SHM_ultData1 M6130 idefine SHM_ultData1 M6131 idefine SHM_ultData12 M6133 idefine SHM_ultData13 M6136 idefine SHM_ultData14 M6136 idefine SHM_ultData13 M6140 idefine SHM_ultData13 M6140 idefine SHM_ultData13 M6140 idefine SHM_ultData20 M6140 idefine	#define SRMB_ultData2	M6124
Hedrine SUMB_litData5 M6127 Hedrine SUMB_litData7 M6128 Hedrine SUMB_litData7 M6129 Hedrine SUMB_litData8 M6130 Hedrine SUMB_litData8 M6131 Hedrine SUMB_litData10 M6132 Hedrine SUMB_litData13 M6133 Hedrine SUMB_litData13 M6133 Hedrine SUMB_litData13 M6139 Hedrine SUMB_litData14 M6139 Hedrine SUMB_litData15 M6139 Hedrine SUMB_litData16 M6139 Hedrine SUMB_litData19 M6141 Hedrine SUMB_litData19 M6142 Hedrine MCSC NSF_RANN M6143 Hedrine MCSC NSF_RANN M6144 Hedrine MCSC NSF_RANN M6145 Hedrine MCSC NSF_RANN M6146 Hedrine MCSC NSF_RANN M6146 Hedrine MCSC NSF_RANN M6148 Hedrine MCSC NSF_RANN M6148 Hedrine MCSC NSF_RANN M6148 Hedrine MCSC NSF_RANN M6151 Hedrine MCSC NSF_RANN M6151 Hedrine MCSC NSF_RANN M6151	#define SRMB_ultData3	M6125
idefine SNM_litbata6 M0128 idefine SNM_litbata8 M0130 idefine SNM_litbata8 M0130 idefine SNM_litbata9 M0131 idefine SNM_litbata1 M0132 idefine SNM_litbata1 M0133 idefine SNM_litbata1 M0134 idefine SNM_litbata1 M0135 idefine SNM_litbata1 M0136 idefine SNM_litbata1 M0137 idefine SNM_litbata1 M0138 idefine SNM_litbata2 M0141 idefine SNM_litbata2 M0143 idefine SNM_SINT COS ACK M0143 idefine SNM_SINT COS CNM M0144 idefine SNM_SINT COS CNM M0143 idefine SNM_SINT COS CNM M0143 idefine SNM_SINT COS CNM M0153 idefine SNM_SINT COS CNM M0154 <td< td=""><td>#define SRMB_ultData4</td><td>M6126</td></td<>	#define SRMB_ultData4	M6126
idefine SNM_litbata6 M0128 idefine SNM_litbata8 M0130 idefine SNM_litbata8 M0130 idefine SNM_litbata9 M0131 idefine SNM_litbata1 M0132 idefine SNM_litbata1 M0133 idefine SNM_litbata1 M0134 idefine SNM_litbata1 M0135 idefine SNM_litbata1 M0136 idefine SNM_litbata1 M0137 idefine SNM_litbata1 M0138 idefine SNM_litbata2 M0141 idefine SNM_litbata2 M0143 idefine SNM_SINT COS ACK M0143 idefine SNM_SINT COS CNM M0144 idefine SNM_SINT COS CNM M0143 idefine SNM_SINT COS CNM M0143 idefine SNM_SINT COS CNM M0153 idefine SNM_SINT COS CNM M0154 <td< td=""><td>#define SRMB_ultData5</td><td>M6127</td></td<>	#define SRMB_ultData5	M6127
idefine SNM_ultData7 M0129 idefine SNM_ultData8 M0131 idefine SNM_ultData1 M0132 idefine SNM_ultData1 M0133 idefine SNM_ultData1 M0134 idefine SNM_ultData1 M0134 idefine SNM_ultData1 M0135 idefine SNM_ultData1 M0137 idefine SNM_ultData1 M0137 idefine SNM_ultData1 M0138 idefine SNM_ultData1 M0141 idefine SNM_ultData2 M0141 idefine SNM_ultData2 M0142 idefine SNM_ultData2 M0150 idefine SNM_ultData2 <td>_</td> <td></td>	_	
idefine SRM_uitbatai M0130 idefine SRM_uitbatai M0131 idefine SRM_uitbatai M0132 idefine SRM_uitbatai M0133 idefine SRM_uitbatai M0134 idefine SRM_uitbatai M0135 idefine SRM_uitbatai M0136 idefine SRM_uitbatai M0136 idefine SRM_uitbatai M0137 idefine SRM_uitbatai M0138 idefine SRM_uitbatai M0138 idefine SRM_uitbatai M0138 idefine SRM_uitbatai M0144 idefine SRC_NST_SRMY M0144 idefine SRC_NST_SRMY M0146 idefine SRC_NST_SRMY M0148 idefine SRC_NST_SRMY M0130 idefine SRC_NST_SRMY M0144 idefine SRC_NST_SRMY M0148 idefine SRC_NST_SRMY M0148 idefine SRC_NST_SRMY M0148 <tr< td=""><td>-</td><td></td></tr<>	-	
idefine SRME_uitbatai M0131 idefine SRME_uitbatai M0132 idefine SRME_uitbatai M0132 idefine SRME_uitbatai M0132 idefine SRME_uitbatai M0133 idefine SRME_uitbatai M0136 idefine SRME_uitbatai M0136 idefine SRME_uitbatai M0136 idefine SRME_uitbatai M0137 idefine SRME_uitbatai M0136 idefine SRME_uitbatai M0136 idefine SRME_uitbatai M0140 idefine SRME_SCONFREXT M0150 idefine SRME_SCONFREXT M0150		
Hedrine SRME PutDatal M6132 Vdofine SRME PutDatal2 M6133 Vdofine SRME PutDatal2 M6134 Vdofine SRME PutDatal3 M6135 Vdofine SRME PutDatal4 M6136 Vdofine SRME PutDatal5 M6137 Vdofine SRME PutDatal6 M6137 Vdofine SRME PutDatal6 M6139 Vdofine SRME PutDatal9 M6141 Vdofine SRME PutDatal9 M6143 Vdofine HCSC NSF SRADY M6144 Vdofine HCSC NSF SRADY M6145 Vdofine HCSC NSF SRADY M6145 Vdofine HCSC NSF SRADY M6146 Vdofine HCSC NSF SRADY M6146 Vdofine HCSC NSF SRADY M6147 Vdofine HCSC NSF SRADY M6137 Vdofine HCSC SSF SRADY M6137 Vdofine HCSC SSF SRADY M6137 Vdofine HCSC SSF SRADY M6138	-	
Hedrine SKMT M0133 Hedrine SKMT Withatal M0134 Hedrine SKMT Withatal M0135 Hedrine SKMT Withatal M0136 Hedrine SKMT Withatal M0137 Hedrine SKMT Withatal M0137 Hedrine SKMT Withatal M0138 Hedrine SKMT Withatal M0139 Hedrine SKMT Withatal M0139 Hedrine SKMT Withatal M0139 Hedrine SKMT Withatal M0141 Hedrine SKMT Withatal M0141 Hedrine HCSC Nor SEADY M0144 Hedrine HCSC Nor SEADY M0150 Hedrine HCSC Nor SEADY M0151 Hedrine HCSC Nor SEADY M0150 Hedrine HCSC Nor SEADY M0153		
Hearine SRAP Tutbaral2 M6134 Vedrine SRAP Tutbaral3 M6135 Vedrine SRAP Tutbaral4 M6136 Vedrine SRAP Tutbaral5 M6137 Vedrine SRAP Tutbaral6 M6138 Vedrine SRAP Tutbaral6 M6139 Vedrine SRAP Tutbaral8 M6140 Vedrine SRAP Tutbaral9 M6141 Vedrine SRAP Tutbaral9 M6143 Vedrine SRAP Tutbaral9 M6143 Vedrine SRAP Tutbaral9 M6144 Vedrine SRAP Tutbaral9 M6143 Vedrine SRAP Tutbaral9 M6143 Vedrine SRAP SRAP M6144 Vedrine HCSC NFF SRAP M6145 Vedrine HCSC NFF SRAP M6146 Vedrine HCSC NFF SRAP M6141 Vedrine HCSC NFF SRAP M6151 Vedrine HCSCC NFF SRAP M6152	—	
Hadrine SNM Tutbatal3 M6135 Hadrine SNM Tutbatal5 M6137 Hadrine SNM Tutbatal5 M6137 Hadrine SNM Tutbatal5 M6138 Hadrine SNM Tutbatal7 M6139 Hadrine SNM Tutbatal7 M6140 Hadrine SNM Tutbatal7 M6140 Hadrine SNM Tutbatal7 M6141 Hadrine SNM Tutbatal7 M6142 Hadrine SCC NST_SERNY M6144 Hadrine KGC NST_SERNY M6144 Hadrine KGC NST_SERNY M6146 Hadrine KGC NST_SERNY M6150 Hadrine KGC NST_SERNY M6151 Hadrine KGC STST_SERNY M6151 Hadrine KGC HST SERNY M6151 Hadrine KGC HST SERNY M6151 Hadrine KGC HST SERNY NMX CMD M6152 Hadrine KGC HST SERNY NMX CMD M6153 Hadrine KGC HST SERNY NMX CMD M6153 Hadrine KGC HST SERNY NMX CMD M6154 Hadrine KGC NST SERNY NMX CMD M6157 Hadrine KGC NST SERNY NMX CMD	-	
idefine SNME_ultbatal4 M6136 idefine SNME_ultbatal5 M6137 idefine SNME_ultbatal6 M6138 idefine SNME_ultbatal7 M6139 idefine SNME_ultbatal9 M6141 idefine SSC_NAVE_Ultbata20 M6142 idefine MCSC_NAVE_NEADY M6143 idefine MCSC_NAVE_NEADY M6144 idefine MCSC_NAVE_NEADY M6145 idefine MCSC_NAVE_NEADY M6145 idefine MCSC_NAVE_NEADY M6146 idefine MCSC_NAVE_NEADY M6147 idefine MCSC_NAVE_NEADY M6147 idefine MCSC_NAVE_NEADY M6147 idefine MCSC_NAVE_NEADY M6148 idefine MCSC_NAVE_NAVE M6149 idefine MCSC_NAVE_NAVE M6153 idefine MCSC_NAVE_NAVE M6154 idefine MCSC_NAVE_NAVE M6154 idefine MCCO_NAVE_NAVE M6154 idefine MCCO_	-	
idefine SHM Libatal5 M6137 idefine SHM Libatal7 M6138 idefine SHM Libatal7 M6138 idefine SHM Libatal7 M6134 idefine SHM Libatal9 M6141 idefine ICSC NSF ISADY M6143 idefine ICSC NSF ISADY M6144 idefine ICSC NSF ISADY M6144 idefine ICSC NSF ISADY M6144 idefine ICSC NSF ISADY M6147 idefine ICSC NSF ISADY M6150 idefine ICSC NSF ISADY M6151 idefine ICSC HSF ISADY M6153 idefine ICSC HSF ISADY M6151 idefine ICSCC HSF ISADY <td>-</td> <td></td>	-	
idefine SRME_ultbatal6 M6138 idefine SRME_ultbatal7 M6139 idefine SRME_ultbatal8 M6140 idefine SRME_ultbatal9 M6141 idefine SRME_ultbatal9 M6143 idefine KSC_NSF_ERAR M6143 idefine KSC_NSF_ERAR M6144 idefine KSC_NSF_ERAR M6145 idefine KSC_NSF_ERAR M6145 idefine KSC_NSF_ERAR M6146 idefine KSC_NSF_ERAR M6147 idefine KSC_NSF_ERAR M6147 idefine KSC_NSF_ERAR M6150 idefine KSC_NSF_ERAR M6151 idefine KSC_NSF_ERAR M6151 idefine KSC_NSF_ERAR M6153 idefine KSC_NSF_ERAR M6154 idefine KSCO_NSF_KSCN MBX_CAR M6153 idefine KSCO_NSF_KSCN MBX_CAR M6164 idefine KSCO_NSF_KSCN MBX_CAR M6164 idefine KSCO_NSF_KSCO_NSK <td></td> <td></td>		
Heafine SRNB_11tbata17 M6139 Heafine SRNB_11tbata19 M6140 Heafine RGSC_NEXPlags M6142 Heafine RGSC_NEXPlags M6143 Heafine RGSC_NEXPLAGS M6144 Heafine RGSC_NEXPLAGS M6144 Heafine RGSC_NEXPLAGS M6146 Heafine RGSC_NEXPLAST COS_ACK M6146 Heafine RGSC_NEXPLAST COS_CMD M6147 Heafine RGSC_NEXPLAST COS_CMD M6147 Heafine RGSC_NEXPLAST COS_CMD M6130 Heafine RGSC_NEXPLAST COS_CMD M6131 Heafine RGSC_STR REXT COS_CMD M6132 Heafine RGSC_NEXPLAST COS_CMD M6135 Heafine RGSC_NEXPLAST COS_CMD M6135 Heafine RGSC_NEXPLAST COS_CMD M6135 Heafine RGSC_NEXPLAST COS_CMD M6135 Heafine RGSC_NEXPLAST COS_CMD M6157 Heafine RGSC_NEXPLAST COS_CMD M6151 Heafine RGSC_NEXPLAST COS_CMD M6161 Heafine RGSC_NEXPLAST COS_CMD M6161 Heafine RGSCO_NGT_RGNT M6161 Heafine RGSCO_NGT_RGNT M6161 Heafine RGSCO_NGT_RGNT M616	#define CDMD withData15	
idefine SKNE_ultData18 M6140 idefine GSC_NNE_stlData20 M6141 idefine GSC_NNE_stlData20 M6143 idefine HGSC_NNE_stlData20 M6143 idefine HGSC_NNE_STLOS M6143 idefine HGSC_NNE_STLOS M6143 idefine HGSC_NNE_STLOS M6147 idefine HGSC_NNE_STLOS M6147 idefine HGSC_NNE_STLOS M6143 idefine HGSC_NNE_STLOS M6143 idefine HGSC_NNE_STLAS M6151 idefine HGSC_NNE_STLAS M6153 idefine HGSC_NNE_STLAS M6154 idefine HGSC_NNE_STLAS M6153 idefine HGSCO_NNE_STLAS M6153 idefine HGSCO_NNE_STLAS M6153 idefine HGSCO_NNE_STLAS M6160 idefine HGSCO_NNE_STLAS M6164 idefine HGSCO_NNE_STLAS M6164 idefine HGSCO_NNE_STLAS M6164 idefine HGSCO_NNE_STLAS		
Hdefine SRME_ultData19 M6141 Hdefine HCSC_DNUTPLAGS M6142 Hdefine HCSC_NNTPLAGS M6143 Hdefine HCSC_NNTPLAGS M6144 Hdefine HCSC_NNTPLAGS M6144 Hdefine HCSC_NNTPLAGS M6144 Hdefine HCSC_NNTPLAGT COS_ACK M6146 Hdefine HCSC_NNTPLAGT COS_CMD M6147 Hdefine HCSC_NTP_SEXC_MDMMX ACK M6148 Hdefine HCSC_NTP_SEXC_MDMX M6150 Hdefine HCSC_NTP_SEXT_COS_CMD M6130 Hdefine HCSC_NTP_REXT_COS_CMD M6153 Hdefine HCSC_NTP_COSTART M6153 Hdefine HCCC_NCY_CMEX_ACK M6156 Hdefine HCCC_NCY_CMEX_ACK M6156 Hdefine HCCC_NCY_CMEX_ACK M6161 Hdefine HCCC_NCY_CMEX_ACK M6161 Hdefine HCCC_NCY_FCCOS_CMM M6161 Hdefine HCCCC_NCY_FCOS_COS_ACK M6162		
Hdefine SKME_iltData20 M6142 Hdefine HCSC_NEX_READY M6143 Hdefine HCSC_NST_READY M6144 Hdefine HCSC_NST_REARC M6145 Hdefine HCSC_NST_COS_ACK M6146 Hdefine HCSC_NST_REATY M6147 Hdefine HCSC_NST_SEND_YEX_COS_CMD M6147 Hdefine HCSC_NST_REATY_COS_CMD M6140 Hdefine HCSC_NST_REATY_COS_CMD M6150 Hdefine HCSC_BRS_REATY M6151 Hdefine HCSC_BRS_REATY_COS_CACK M6151 Hdefine HCSC_BRS_REATY_COS_CACK M6152 Hdefine HCSC_BRS_REATY_COS_CACK M6153 Hdefine HCSC_BRS_REATY_COS_CACK M6154 Hdefine HCCCO_NCF_BREATY_COS_CACK M6155 Hdefine HCCCO_NCF_BREATY_COS_CACK M6156 Hdefine HCCCO_NCF_BREATY_COS_CACK M6150 Hdefine HCCCO_NCF_FERROR M6150 Hdefine HCCCO_NCF_BREATY_COS_CACK M6161 Hdefine HCCCO_NCF_FERROR M6161 Hdefine HCCCO_NCF_POD_TN_CMD M6163 Hdefine HCCCO_NCF_FERROR M6160 Hdefine HCCCO_NCF_POD_TN_CMD M6164 Hdefine HC	-	
idefine HCSC DNFLXFlags M6143 idefine HCSC NSF_ERROR M6144 idefine HCSC NSF_ERROR M6145 idefine HCSC NSF_ERROR M6145 idefine HCSC NSF_ERROR M6146 idefine HCSC NSF_ERROR M6147 idefine HCSC NSF_ERROR/MSX_CMD M6149 idefine HCSC HSF_ERSFT M6151 idefine HCSC HSF_ERSFT M6153 idefine HCSC HSF_ERSFT M6154 idefine HCSC HSF_ERSFT M6154 idefine HCCSC HSF_ERSFT M6154 idefine HCCCO NCC COMMUNICATING M6156 idefine HCCCO NCC HST_ERSR M6161 idefine HCCCO NCC HST_ECS CMD M6163 idefine HCCCO NCC HST_ECS CMD M6163 idefine HCCCO NCC HST_ECS CMD M6166 idefine HCCCO NCC HST_ECS CMD M6161 idefine HCCCO NCC HST_ECS CMD M6161 <	-	
idefine HCSC_NSF_READY M6144 idefine HCSC_NSF_REARA M6145 idefine HCSC_NSF_HORT_COS_ACK M6146 idefine HCSC_NSF_SEND_MEX_ACK M6147 idefine HCSC_NSF_SEND_MEX_ACK M6148 idefine HCSC_NSF_REACY_MEX_CMD M6149 idefine HCSC_NSF_REACY_MEX_CMD M6150 idefine HCSC_NSF_REACY_MEX_CMD M6151 idefine HCSC_NSF_REACY_MEX_CMD M6151 idefine HCSC_NSF_SEND_MEX_CCM M6152 idefine HCSC_NSF_SEND_MEX_COS_ACK M6154 idefine HCSC_NSF_SEND_MEX_CMD M6155 idefine HCCCO_NCF_COMMUNICATING M6156 idefine HCCCO_NCF_ROMUNICATING M6161 idefine HCCCO_NCF_ROMUNICATING M6161 idefine HCCCO_NCF_ROMUNICATING M6163 idefine HCCCO_NCF_ROMUNICATING M6163 idefine HCCCO_NCF_ROMUNICATING M6164 idefine HCCCO_NCF_ROMUNICATING M6163 idefine HCCCO_NCF_ROMUNICATING M6163 idefine HCCCO_NCF_ROMUNICATING M6163 idefine HCCCO_NCF_ROMUNICATING M6164 idefine HCCCO_NCF_ROMUNICATING M6164		
Hadrine HCSC NSF DERXCR M6145 Hdefine HCSC NSF NETX COS CMD M6147 Hdefine HCSC NSF NETX COS CMD M6147 Hdefine HCSC NSF STERV_MBX_CMD M6148 Hdefine HCSC NSF STERV_MBX_CMD M6150 Hdefine HCSC STS FSTERV M6151 Hdefine HCSC STR STERVE M6151 Hdefine HCSC SC STR FOR COS CMD M6152 Hdefine HCSC SC STR FOR COS CMD M6153 Hdefine HCSC STR STR STR COS CMD M6154 Hdefine HCSC STR STR STR CMS CMD M6155 Hdefine HCCC USTECTTARS M6156 Hdefine HCCC ONCE COMMUNICATING M6157 Hdefine HCCC ONCE STR SRO M6161 Hdefine HCCC ONCE STR SRO M6161 Hdefine HCCC ONCE STR SRO M6163 Hdefine HCCC ONCE STR SRO M6163 Hdefine HCCC ONCE STR MAX CMD M6164 Hdefine HCCC ONCE STR MAX CMD M6166 Hdefine HCCC ONCE STR MAX CMD M6166 Hdefine HCCC ONCE STR MAX CMD M6166 Hdefine HCCC ONCE STR MAX CMD M6167 Hdefine HCCC ONCE STR MAX CMD M6166 Hd		
#define HCSC_NSF_HOST_COS_ACK M6146 #define HCSC_NSF_END_MEX_ACK M6147 #define HCSC_NSF_SEND_MEX_ACK M6149 #define HCSC_NSF_REV_MEX_CMD M6150 #define HCSC_BF_REST M6150 #define HCSC_HSF_REST M6151 #define HCSC_HSF_REST M6152 #define HCSC_HSF_REST M6154 #define HCSC_HSF_REST_COS_ACK M6154 #define HCSC_HSF_REST_MO_MSX_CMD M6155 #define HCCCO_NCF_COMMUNICATING M6158 #define HCCO_NCF_COMMUNICATING M6161 #define HCCCO_NCF_END_MSX_CCMD M6162 #define HCCCO_NCF_END_MSX_CCMD M6163 #define HCCCO_NCF_END_MSX_CCMD M6161 #define HCCCO_NCF_END_MSX_CCMD M6161 #define HCCCO_NCF_END_MSX_CCMD M6161		
#define HCSC NSF_NETX_COG_CMD M6147 #define HCSC NSF_NECV_MEX_CMD M6148 #define HCSC NSF_NECV_MEX_CMD M6150 #define HCSC_NSF_NECV_MEX_CMD M6150 #define HCSC_NSF_NECV_MEX_CMD M6151 #define HCSC_HSF_NECV_MEX_CMD M6151 #define HCSC_HSF_NECV_MEX_CMD M6153 #define HCSC_HSF_NECV_MEX_ACK M6155 #define HCSC_NFF_NECV_MEX_ACK M6155 #define HCCC0_NCF_COMMUNICATING M6158 #define HCCC0_NCF_NETX_COS_CMD M6161 #define HCCC0_NCF_NETX_COS_CMD M6161 #define HCCC0_NCF_NETX_COS_CMD M6161 #define HCCC0_NCF_NETX_COS_CMD M6162 #define HCCC0_NCF_NETX_COS_CMD M6164 #define HCCC0_NCF_PD0_ID_UT_ACK M6166 #define HCCC0_NCF_PD1_NCMD M6167 #define HCCC0_NCF_PD1_NCMD M6167 #define HCCC0_NCF_PD1_NCMD M6167 #define HCCC0_NCF_PD1_NCMD M6167 #define HCCC0_NCF_PD0_ID_NCKA M6170 #define HCCC0_NCF_PD0_ID_NCKA M6171 #define HCCC0_NCF_PD0_ID_NCKA M6171		
#define HCSC_NFF_SEND_MBX_ACKM6148#define HCSC_NFF_SEND_MBX_ACKM6150#define HCSC_NFF_SEND_MBX_TM6151#define HCSC_NFF_NESTM6152#define HCSC_NFF_NESTM6152#define HCSC_NFF_NEST_COS_ACKM6154#define HCSC_NFF_SEND_MBX_CMDM6155#define HCCC_NFF_NEST_COS_ACKM6156#define HCCC_NFF_NEST_COS_ACKM6156#define HCCC_NFF_NEST_COS_ACKM6156#define HCCC_NFF_NEST_COS_ACKM6157#define HCCC_NFF_NEST_COS_ACKM6161#define HCCCO_NCF_DENT_COS_CMDM6161#define HCCCO_NCF_NEST_COS_ACKM6161#define HCCCO_NCF_NEST_COS_ACKM6161#define HCCCO_NCF_PENT_COS_ACKM6163#define HCCCO_NCF_PENT_COS_CMDM6163#define HCCCO_NCF_PDI_NCMDM6164#define HCCCO_NCF_PDI_NCMDM6166#define HCCCO_NCF_PDI_NCMDM6166#define HCCCO_NCF_PDI_NCMDM6170#define HCCCO_NCF_PDI_NCMDM6171#define HCCCO_NCF_PDO_OUT_ACKM6173#define HCCCO_NCF_PDO_OUT_CMDM6173#define HCCCO_NCF_REST_MS_ACKM6174#define HCCCO_NCF_REST_MS_ACKM6176#define HCCCO_NCF_REST_MS_ACKM6176#define HCCCO_NCF_REST_MS_ACKM6176#define HCCCI_NCF_REST_MS_ACKM6180#define HCCCI_NCF_REST_MS_ACKM6181#define HCCCI_NCF_REST_MS_ACKM6180#define HCCCI_NCF_REST_MS_ACKM6182#define HCCCI_NCF_REST_MS_ACKM6182#define HCCCI_NCF_REST_MS_ACKM6182 </td <td>#aerine HCSC_NSF_HOST_COS_ACK</td> <td></td>	#aerine HCSC_NSF_HOST_COS_ACK	
Hefine HCSC_NSF_RECV_MSX_CMDM6149Hefine HCSC_HSF_RESETM6150Hefine HCSC_HSF_RESETM6151Hefine HCSC_HSF_RESETM6152Hefine HCSC_HSF_ROT_COS_CMDM6153Hefine HCSC_HSF_NETX_COS_ACKM6154Hefine HCSC_HSF_RECV_MSX_ACKM6155Hefine HCCC0_USPECTURIATARM6156Hefine HCCC0_USPECTURIATARM6156Hefine HCCC0_NCF_CMNUNICATINGM6158Hefine HCCC0_NCF_RERORM6159Hefine HCCC0_NCF_REND_MSX_ACKM6160Hefine HCCC0_NCF_REND_MSX_ACKM6161Hefine HCCC0_NCF_REND_MSX_ACKM6162Hefine HCCC0_NCF_REND_MSX_ACKM6164Hefine HCCC0_NCF_REND_MSX_ACKM6164Hefine HCCC0_NCF_REND_MSX_ACKM6164Hefine HCCC0_NCF_PD_ID_OTT_ACKM6166Hefine HCCC0_NCF_PD_ID_OTT_ACKM6166Hefine HCCC0_HCF_HOST_COS_ACKM6170Hefine HCCC0_HCF_HOST_COS_ACKM6170Hefine HCCC0_HCF_RENT_COS_ACKM6170Hefine HCCC0_HCF_RENT_COS_ACKM6170Hefine HCCC0_HCF_PD_ID_OTT_CMDM6173Hefine HCCC0_HCF_PD_ID_OTT_CMDM6173Hefine HCCC1_HCF_REOV_MSX_ACKM6174Hefine HCCC1_NCF_REND_MSX_ACKM6174Hefine HCCC1_NCF_REND_MSX_ACKM6176Hefine HCCC1_NCF_REND_MSX_ACKM6176Hefine HCCC1_NCF_REND_MSX_ACKM6176Hefine HCCC1_NCF_REND_MSX_ACKM6176Hefine HCCC1_NCF_REND_MSX_ACKM6176Hefine HCCC1_NCF_REND_MSX_ACKM6176Hefine HCCC1_NCF_REND_MSX_ACKM6176 <t< td=""><td></td><td></td></t<>		
Hearine HCSC_DHostPlagsM6150Hearine HCSC_HSF_BSETM6151Hearine HCSC_HSF_BOUTSTARTM6152Hearine HCSC_HSF_BOUTSTARTM6153Hearine HCSC_HSF_SEND_MSA_CMDM6154Hearine HCSC_HSF_SEND_MSA_CMDM6154Hearine HCSC_HSF_SEND_MSA_CMDM6156Hearine HCCC_UNCY_COMMUNICATINGM6156Hearine HCCCO_NCF_ORMUNICATINGM6161Hearine HCCCO_NCF_ORMUNICATINGM6161Hearine HCCCO_NCF_NETX_COS_CMDM6161Hearine HCCCO_NCF_NETX_COS_CMDM6161Hearine HCCCO_NCF_DUT_ACKM6163Hearine HCCCO_NCF_PDO_UT_ACKM6164Hearine HCCCO_NCF_PDO_UT_ACKM6165Hearine HCCCO_NCF_PDI_UT_ACKM6166Hearine HCCCO_NCF_PDI_UT_ACKM6166Hearine HCCCO_NCF_PDI_UT_ACKM6167Hearine HCCCO_UNFPU_UT_ACKM6168Hearine HCCCO_UNF_PDI_UT_ACKM6167Hearine HCCCO_UNF_PDI_TN_CMDM6167Hearine HCCCO_UNF_PDI_TN_CMDM6167Hearine HCCCO_UNF_PDI_TN_CMDM6170Hearine HCCCO_HCF_PDO_UT_CMDM6173Hearine HCCCO_HCF_PDO_UT_CMDM6174Hearine HCCCI_HCF_PDO_UT_CMDM6175Hearine HCCCI_HCF_PDI_UT_CMDM6174Hearine HCCCI_NCF_RERN_MAX_CMM6170Hearine HCCCI_NCF_RERN_MAX_CMM6174Hearine HCCCI_NCF_RERN_MAX_CMM6174Hearine HCCCI_NCF_RERN_MAX_CMM6176Hearine HCCCI_NCF_RERN_MAX_CMM6181Hearine HCCCI_NCF_RERN_MAX_CMM6181Hearine HCCCI_NCF_RERN_MAX_CMM6181 <td></td> <td></td>		
#define HCSC_HSF_HOST_COS_CMDM6153#define HCSC_HSF_ENT_COS_CACKM6154#define HCSC_HSF_RENT_COS_ACKM6155#define HCCC0_USNETXFlagsM6157#define HCCC0_USNETXFLAGSM6157#define HCCC0_NCF_DST_COS_ACKM6160#define HCCC0_NCF_SEND_MBX_ACKM6161#define HCCC0_NCF_PDI_TACS_CMDM6163#define HCCC0_NCF_PDI_TACKM6164#define HCCC0_NCF_PDI_TACKM6165#define HCCC0_NCF_PDI_TACKM6166#define HCCC0_NCF_PDI_TACKM6166#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6171#define HCCC0_NCF_PDI_TACKM6171#define HCCC0_NCF_PDI_TACKM6172#define HCCC0_NCF_PDI_TACKM6172#define HCCC0_NCF_PDI_TACKM6172#define HCCC0_NCF_PDI_TACKM6173#define HCCC0_NCF_PDI_TACKM6173#define HCCC0_NCF_PDI_TACKM6174#define HCCC1_NCF_RECV_MBX_CMDM6173#define HCCC1_NCF_NONT_CANTINGM6173#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174	#define HCSC_NSF_RECV_MBX_CMD	
#define HCSC_HSF_HOST_COS_CMDM6153#define HCSC_HSF_ENT_COS_CACKM6154#define HCSC_HSF_RENT_COS_ACKM6155#define HCCC0_USNETXFlagsM6157#define HCCC0_USNETXFLAGSM6157#define HCCC0_NCF_DST_COS_ACKM6160#define HCCC0_NCF_SEND_MBX_ACKM6161#define HCCC0_NCF_PDI_TACS_CMDM6163#define HCCC0_NCF_PDI_TACKM6164#define HCCC0_NCF_PDI_TACKM6165#define HCCC0_NCF_PDI_TACKM6166#define HCCC0_NCF_PDI_TACKM6166#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6171#define HCCC0_NCF_PDI_TACKM6171#define HCCC0_NCF_PDI_TACKM6172#define HCCC0_NCF_PDI_TACKM6172#define HCCC0_NCF_PDI_TACKM6172#define HCCC0_NCF_PDI_TACKM6173#define HCCC0_NCF_PDI_TACKM6173#define HCCC0_NCF_PDI_TACKM6174#define HCCC1_NCF_RECV_MBX_CMDM6173#define HCCC1_NCF_NONT_CANTINGM6173#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174	#define HCSC_bHostFlags	
#define HCSC_HSF_HOST_COS_CMDM6153#define HCSC_HSF_ENT_COS_CACKM6154#define HCSC_HSF_RENT_COS_ACKM6155#define HCCC0_USNETXFlagsM6157#define HCCC0_USNETXFLAGSM6157#define HCCC0_NCF_DST_COS_ACKM6160#define HCCC0_NCF_SEND_MBX_ACKM6161#define HCCC0_NCF_PDI_TACS_CMDM6163#define HCCC0_NCF_PDI_TACKM6164#define HCCC0_NCF_PDI_TACKM6165#define HCCC0_NCF_PDI_TACKM6166#define HCCC0_NCF_PDI_TACKM6166#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6167#define HCCC0_NCF_PDI_TACKM6171#define HCCC0_NCF_PDI_TACKM6171#define HCCC0_NCF_PDI_TACKM6172#define HCCC0_NCF_PDI_TACKM6172#define HCCC0_NCF_PDI_TACKM6172#define HCCC0_NCF_PDI_TACKM6173#define HCCC0_NCF_PDI_TACKM6173#define HCCC0_NCF_PDI_TACKM6174#define HCCC1_NCF_RECV_MBX_CMDM6173#define HCCC1_NCF_NONT_CANTINGM6173#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174#define HCCC1_NCF_NONT_CANTINGM6174	#define HCSC_HSF_RESET	
HetineHCSC_HSF_NETX_COS_ACKM6154#defineHCSC_HSF_SEND_MEX_CMDM6155#defineHCCC_HSF_RECV_MEX_ACKM6156#defineHCCCO_USNEtXFlagsM6157#defineHCCCO_NCF_RERORM6159#defineHCCCO_NCF_NETX_COS_CMDM6160#defineHCCCO_NCF_NETX_COS_CMDM6161#defineHCCCO_NCF_PECV_MEX_ACKM6162#defineHCCCO_NCF_PECV_MEX_COS_CMDM6163#defineHCCCO_NCF_PECV_MEX_CMDM6163#defineHCCCO_NCF_PD_OUT_ACKM6166#defineHCCCO_NCF_PD_OUT_ACKM6166#defineHCCCO_NCF_PD_OUT_ACKM6167#defineHCCCO_NCF_PD_OUT_ACKM6167#defineHCCCO_NCF_PD_OUT_ACKM6169#defineHCCCO_HCF_NETX_COS_ACKM6170#defineHCCCO_HCF_NETX_COS_ACKM6170#defineHCCCO_HCF_NETX_COS_ACKM6171#defineHCCCO_HCF_PD_UT_ACKM6172#defineHCCCO_HCF_PD_UT_ACKM6173#defineHCCCO_HCF_PD_UT_ACKM6174#defineHCCCO_HCF_PD_UT_ACKM6176#defineHCCCI_NCF_ERORM6177#defineHCCCI_NCF_ERORM6177#defineHCCCI_NCF_ERORM6173#defineHCCCI_NCF_ERORM6173#defineHCCCI_NCF_ERORM6174#defineHCCCI_NCF_ERORM6174#defineHCCCI_NCF_ERORM6173#defineHCCCI_NCF_ERORM6176#defineHCCCI_NCF_ER	#define HCSC_HSF_BOOTSTART	
#defineHCSC_HSF_RECV_MBX_ACKM6155#defineHCCC0_USNetxFlagsM6157#defineHCCC0_USNetxFlagsM6159#defineHCCC0_NCF_COMUNICATINGM6159#defineHCCC0_NCF_ERRORM6160#defineHCCC0_NCF_BSTD_COS_ACKM6161#defineHCCC0_NCF_SEND_MBX_ACKM6161#defineHCCC0_NCF_PDO_TACKM6163#defineHCCC0_NCF_PDO_TACKM6164#defineHCCC0_NCF_PDO_TACKM6166#defineHCCC0_NCF_PD_IT_CMDM6167#defineHCCC0_NCF_PDI_TACKM6168#defineHCCC0_HCF_PDI_TACKM6167#defineHCCC0_HCF_PDI_TACKM6167#defineHCCC0_HCF_PDI_TACKM6171#defineHCCC0_HCF_PDI_TACKM6172#defineHCCC0_HCF_PDI_TACKM6172#defineHCCC0_HCF_PDI_TACKM6172#defineHCC0_HCF_PDI_TACKM6173#defineHCC0_HCF_PDI_TACKM6174#defineHCC0_HCF_PDI_TACKM6176#defineHCC1_HCF_PDI_TACKM6176#defineHCC1_NCF_ERRORM6179#defineHCC1_NCF_ERRORM6180#defineHCC1_NCF_ERRORM6181#defineHCC1_NCF_ERRORM6183#defineHCC1_NCF_PD_OUT_ACKM6183#defineHCC1_NCF_PD_OUT_ACKM6183#defineHCC1_NCF_PD_UT_ACKM6183#defineHCC1_NCF_PD_UT_ACKM6184#defineHCC1_NCF_PD_UT_ACKM6184 <td></td> <td></td>		
#defineHCSC_USE_REC_MEX_ACKM6156#defineHCCCO_USE_REC_PLAGSM6157#defineHCCCO_USE_RECRM6159#defineHCCCO_USE_RECRM6160#defineHCCCO_USE_RECRM6161#defineHCCCO_USE_RECX_BXA_CKM6162#defineHCCCO_USE_RECY_BXA_CKM6163#defineHCCCO_USE_RECY_BXA_CKM6163#defineHCCCO_USE_PDI_UT_ACKM6166#defineHCCCO_USE_PDI_UT_ACKM6166#defineHCCCO_USE_PDI_UT_ACKM6166#defineHCCCO_USE_RECY_BXA_CKM6166#defineHCCCO_USE_RECY_BXA_CKM6167#defineHCCCO_USE_RECY_BXA_CKM6170#defineHCCCO_USE_RECY_MXA_CKM6171#defineHCCCO_USE_RECY_MXA_CKM6173#defineHCCCO_USE_RECY_MXA_CKM6173#defineHCCCO_USE_RECY_MXA_CKM6174#defineHCCCO_USE_RECY_MXA_CKM6176#defineHCCCU_USE_RECY_MXA_CKM6176#defineHCCCU_USE_RECY_MXA_CKM6176#defineHCCCU_USE_RECY_MXA_CKM6180#defineHCCCU_USE_RECY_MXA_CKM6180#defineHCCCU_USE_RECY_MXA_CKM6180#defineHCCCU_USE_RECY_MXA_CKM6180#defineHCCCU_USE_RECY_MXA_CKM6180#defineHCCCU_USE_RECY_MXA_CKM6180#defineHCCCU_USE_RECY_MXA_CKM6180#defineHCCCU_USE_RECY_MXA_CKM6180#defineHCCCU_USE_RECY_MXA_CKM6180		
#defineHCCCD_usNetxFlagsM6157#defineHCCCD_NCF_CMMUNICATINGM6158#defineHCCCD_NCF_ERRORM6159#defineHCCCD_NCF_METX_COS_CMDM6160#defineHCCCD_NCF_SEND_MEX_ACKM6161#defineHCCCD_NCF_SEND_MEX_ACKM6163#defineHCCCD_NCF_PDO_IT_ACKM6164#defineHCCCD_NCF_PDO_IT_ACKM6166#defineHCCCD_USF_PDI_IT_CMDM6167#defineHCCCD_USF_PDI_IT_CMDM6168#defineHCCCD_USF_SEND_MEX_ACKM6168#defineHCCCD_USF_SEND_MEX_CMDM6171#defineHCCCD_USF_SEND_MEX_CMDM6171#defineHCCCD_HCF_SEND_MEX_CMDM6172#defineHCCCD_HCF_SEND_MEX_CMDM6173#defineHCCCD_HCF_PDO_OUT_CMDM6173#defineHCCCD_HCF_PDO_OUT_CMDM6176#defineHCCCD_HCF_PDI_TN_ACKM6176#defineHCCCL_NCF_ERORM6177#defineHCCCL_NCF_ERORM6178#defineHCCCL_NCF_ERORM6179#defineHCCL_NCF_ERORM6180#defineHCCL_NCF_ERORM6180#defineHCCL_NCF_ERON_MEX_ACKM6182#defineHCCL_NCF_ERON_MEX_ACKM6183#defineHCCL_NCF_ERON_MEX_ACKM6183#defineHCCL_NCF_ERON_MEX_ACKM6183#defineHCCL_NCF_ERON_MEX_ACKM6184#defineHCCL_NCF_ERON_MEX_ACKM6184#defineHCCL_NCF_ERON_MEX_ACKM6186#de		
#defineHCCCO_NCF_ERRORM6159#defineHCCCO_NCF_NETX_COS_ACKM6160#defineHCCCO_NCF_NETX_COS_CMDM6161#defineHCCCO_NCF_RECV_MEX_CMDM6162#defineHCCCO_NCF_PDO_OUT_ACKM6163#defineHCCCO_NCF_PDO_OUT_ACKM6166#defineHCCCO_NCF_PDI_N_CMDM6166#defineHCCCO_NCF_PDI_N_CMDM6167#defineHCCCO_NCF_PDI_N_CMDM6168#defineHCCCO_USHOSTPLAGSM6169#defineHCCCO_USHOSTPLAGSM6170#defineHCCCO_HCF_SEND_MEX_CMDM6171#defineHCCCO_HCF_RECV_MEX_ACKM6172#defineHCCCO_HCF_PDO_OUT_CMDM6173#defineHCCCO_HCF_PDO_IN_ACKM6174#defineHCCCI_NCF_PDO_IN_ACKM6176#defineHCCCI_NCF_COMUNICATINGM6177#defineHCCCI_NCF_COMUNICATINGM6178#defineHCCCI_NCF_ERRORM6179#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_NETX_COS_CMDM6182#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_NETX_COS_CMDM6186#defineHCCCI_NCF_NETX_COS_CMDM6187#defineHCCCI_NCF_NETX_COS_CMDM6187#defineHCCCI_NCF_NETX_COS_CMD </td <td></td> <td></td>		
#defineHCCCO_NCF_ERRORM6159#defineHCCCO_NCF_NETX_COS_ACKM6160#defineHCCCO_NCF_NETX_COS_CMDM6161#defineHCCCO_NCF_RECV_MEX_CMDM6162#defineHCCCO_NCF_PDO_OUT_ACKM6163#defineHCCCO_NCF_PDO_OUT_ACKM6166#defineHCCCO_NCF_PDI_N_CMDM6166#defineHCCCO_NCF_PDI_N_CMDM6167#defineHCCCO_NCF_PDI_N_CMDM6168#defineHCCCO_USHOSTPLAGSM6169#defineHCCCO_USHOSTPLAGSM6170#defineHCCCO_HCF_SEND_MEX_CMDM6171#defineHCCCO_HCF_RECV_MEX_ACKM6172#defineHCCCO_HCF_PDO_OUT_CMDM6173#defineHCCCO_HCF_PDO_IN_ACKM6174#defineHCCCI_NCF_PDO_IN_ACKM6176#defineHCCCI_NCF_COMUNICATINGM6177#defineHCCCI_NCF_COMUNICATINGM6178#defineHCCCI_NCF_ERRORM6179#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_NETX_COS_CMDM6182#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_NETX_COS_CMDM6186#defineHCCCI_NCF_NETX_COS_CMDM6187#defineHCCCI_NCF_NETX_COS_CMDM6187#defineHCCCI_NCF_NETX_COS_CMD </td <td>#define HCCCO_usNetxFlags</td> <td></td>	#define HCCCO_usNetxFlags	
#defineHCCC0_NCF_HOST_COS_ACKM6160#defineHCCC0_NCF_NETX_COS_CMDM6161#defineHCCC0_NCF_SEND_MBX_ACKM6162#defineHCCC0_NCF_SEND_MBX_ACKM6163#defineHCCC0_NCF_PD0_OUT_ACKM6165#defineHCCC0_NCF_PD1_OUT_ACKM6166#defineHCCC0_NCF_PD1_IN_CMDM6167#defineHCCC0_HCF_NETX_COS_ACKM6170#defineHCCC0_HCF_NETX_COS_ACKM6171#defineHCCC0_HCF_NETX_COS_ACKM6171#defineHCCC0_HCF_SEND_MBX_CMDM6171#defineHCCC0_HCF_SEND_MSX_CMDM6172#defineHCCC0_HCF_PD0_OUT_CMDM6173#defineHCCC0_HCF_PD0_OUT_CMDM6174#defineHCCC0_HCF_PD1_IN_ACKM6176#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_ERRORM6180#defineHCCC1_NCF_ERRORM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_ERRORM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_NETX_COS_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6183#defineHCCC1_NCF_PD0_OUT_ACKM6186#defineHCCC1_NCF_PD0_OUT_ACKM6186#defineHCCC1_NCF_PD0_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_ACK </td <td></td> <td></td>		
#defineHCCCO_NCE_PDD_OUT_ACKM6164#defineHCCCO_NCE_PDI_IN_CMDM6165#defineHCCCO_NCE_PDI_IN_CMDM6167#defineHCCCO_USHOSTLAGSM6168#defineHCCCO_HCF_NETX_COS_ACKM6170#defineHCCCO_HCF_NETX_COS_ACKM6171#defineHCCCO_HCF_RECV_MBX_CMDM6173#defineHCCCO_HCF_PDO_IN_ACKM6173#defineHCCCO_HCF_PDO_IN_ACKM6174#defineHCCCO_HCF_PDI_IN_ACKM6176#defineHCCCI_NCF_PDI_IN_ACKM6177#defineHCCCI_NCF_COMMUNICATINGM6178#defineHCCCI_NCF_COMMUNICATINGM6181#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_ERORM6181#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_PDI_UTX_CMDM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACK<	#define HCCCU_NCF_ERROR	
#defineHCCCO_NCE_PDD_OUT_ACKM6164#defineHCCCO_NCE_PDI_IN_CMDM6165#defineHCCCO_NCE_PDI_IN_CMDM6167#defineHCCCO_USHOSTLAGSM6168#defineHCCCO_HCF_NETX_COS_ACKM6170#defineHCCCO_HCF_NETX_COS_ACKM6171#defineHCCCO_HCF_RECV_MBX_CMDM6173#defineHCCCO_HCF_PDO_IN_ACKM6173#defineHCCCO_HCF_PDO_IN_ACKM6174#defineHCCCO_HCF_PDI_IN_ACKM6176#defineHCCCI_NCF_PDI_IN_ACKM6177#defineHCCCI_NCF_COMMUNICATINGM6178#defineHCCCI_NCF_COMMUNICATINGM6181#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_ERORM6181#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_PDI_UTX_CMDM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACK<	#define HCCCU_NCF_HOST_COS_ACK	
#defineHCCCO_NCE_PDD_OUT_ACKM6164#defineHCCCO_NCE_PDI_IN_CMDM6165#defineHCCCO_NCE_PDI_IN_CMDM6167#defineHCCCO_USHOSTLAGSM6168#defineHCCCO_HCF_NETX_COS_ACKM6170#defineHCCCO_HCF_NETX_COS_ACKM6171#defineHCCCO_HCF_RECV_MBX_CMDM6173#defineHCCCO_HCF_PDO_IN_ACKM6173#defineHCCCO_HCF_PDO_IN_ACKM6174#defineHCCCO_HCF_PDI_IN_ACKM6176#defineHCCCI_NCF_PDI_IN_ACKM6177#defineHCCCI_NCF_COMMUNICATINGM6178#defineHCCCI_NCF_COMMUNICATINGM6181#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_ERORM6181#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_PDI_UTX_CMDM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACK<	#define HCCCU_NCF_NETX_COS_CMD	
#defineHCCCO_NCE_PDD_OUT_ACKM6164#defineHCCCO_NCE_PDI_IN_CMDM6165#defineHCCCO_NCE_PDI_IN_CMDM6167#defineHCCCO_USHOSTLAGSM6168#defineHCCCO_HCF_NETX_COS_ACKM6170#defineHCCCO_HCF_NETX_COS_ACKM6171#defineHCCCO_HCF_RECV_MBX_CMDM6173#defineHCCCO_HCF_PDO_IN_ACKM6173#defineHCCCO_HCF_PDO_IN_ACKM6174#defineHCCCO_HCF_PDI_IN_ACKM6176#defineHCCCI_NCF_PDI_IN_ACKM6177#defineHCCCI_NCF_COMMUNICATINGM6178#defineHCCCI_NCF_COMMUNICATINGM6181#defineHCCCI_NCF_NETX_COS_CMDM6181#defineHCCCI_NCF_ERORM6181#defineHCCCI_NCF_NETX_COS_CMDM6183#defineHCCCI_NCF_PDI_UTX_CMDM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6183#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACKM6186#defineHCCCI_NCF_PDI_OTT_ACK<	#define HCCCU_NCF_SEND_MBX_ACK	
#defineHCCC0_NCF_PD1_UN_CMDM6167#defineHCCC0_uSHOstFlagsM6168#defineHCCC0_HCF_NST_COS_CMDM6169#defineHCCC0_HCF_NETX_COS_ACKM6170#defineHCCC0_HCF_SEND_MBX_CMDM6171#defineHCCC0_HCF_RECV_MBX_ACKM6172#defineHCCC0_HCF_PD0_OUT_CMDM6173#defineHCCC0_HCF_PD1_OUT_CMDM6173#defineHCCC0_HCF_PD1_OUT_CMDM6175#defineHCCC0_HCF_PD1_OUT_CMDM6176#defineHCCC1_USNetrlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_DERCQM6181#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_NETX_COS_CMDM6182#defineHCCC1_NCF_RECV_MBX_ACKM6182#defineHCCC1_NCF_PD0_OUT_ACKM6183#defineHCCC1_NCF_PD0_OUT_ACKM6186#defineHCCC1_NCF_PD1_UT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6188#defineHCCC1_HCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6188#defineHCC1_HCF_HOST_COS_CMDM6188	#define HCCCU_NCF_RECV_MBX_CMD	
#defineHCCC0_NCF_PD1_UN_CMDM6167#defineHCCC0_uSHOstFlagsM6168#defineHCCC0_HCF_NST_COS_CMDM6169#defineHCCC0_HCF_NETX_COS_ACKM6170#defineHCCC0_HCF_SEND_MBX_CMDM6171#defineHCCC0_HCF_RECV_MBX_ACKM6172#defineHCCC0_HCF_PD0_OUT_CMDM6173#defineHCCC0_HCF_PD1_OUT_CMDM6173#defineHCCC0_HCF_PD1_OUT_CMDM6175#defineHCCC0_HCF_PD1_OUT_CMDM6176#defineHCCC1_USNetrlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_DERCQM6181#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_NETX_COS_CMDM6182#defineHCCC1_NCF_RECV_MBX_ACKM6182#defineHCCC1_NCF_PD0_OUT_ACKM6183#defineHCCC1_NCF_PD0_OUT_ACKM6186#defineHCCC1_NCF_PD1_UT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6188#defineHCCC1_HCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6187#defineHCCC1_HCF_PD1_IN_CMDM6188#defineHCC1_HCF_HOST_COS_CMDM6188	#aeiine HCCCU_NCF_PDU_OUT_ACK	
#defineHCCC0_NCF_PD1_IN_CMDM6167#defineHCCC0_usHostFlagsM6168#defineHCCC0_HCF_HOST_COS_CMDM6169#defineHCCC0_HCF_NEXT_COS_ACKM6170#defineHCCC0_HCF_SEND_MBX_CMDM6171#defineHCCC0_HCF_RECV_MBX_ACKM6172#defineHCCC0_HCF_PD0_OUT_CMDM6173#defineHCCC0_HCF_PD1_UN_ACKM6175#defineHCCC0_HCF_PD1_IN_ACKM6176#defineHCCC1_USNetxFlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MBX_ACKM6182#defineHCCC1_NCF_RECV_MBX_ACKM6182#defineHCCC1_NCF_PD0_UT_ACKM6183#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD0_IN_CMDM6186#defineHCCC1_NCF_PD1_IN_CMDM6186#defineHCCC1_NCF_PD1_IN_CMDM6186#defineHCCC1_NCF_PD1_IN_CMDM6186#defineHCCC1_NCF_PD1_IN_CMDM6186#defineHCCC1_NCF_PD1_IN_CMDM6186#defineHCCC1_HCF_PD1_IN_CMDM6186#defineHCCC1_HCF_PD1_IN_CMDM6188#defineHCCC1_HCF_PD1_IN_CMDM6188#defineHCCC1_HCF_HOST_COS_CMDM6189	#aeine HCCCU_NCF_PDU_IN_CMD	
#defineHCCCU_USHOSTFlagsM6168#defineHCCCU_HCF_HOST_COS_CMDM6169#defineHCCCU_HCF_SEND_MBX_CMDM6171#defineHCCCU_HCF_SEND_MBX_CMDM6171#defineHCCCU_HCF_PDO_OUT_CMDM6172#defineHCCCU_HCF_PDO_IN_ACKM6173#defineHCCCU_HCF_PDI_OUT_CMDM6175#defineHCCCL_HCF_PDI_OUT_CMDM6176#defineHCCCL_USNetxFlagsM6177#defineHCCCL_NCF_COMMUNICATINGM6178#defineHCCCL_NCF_HOST_COS_ACKM6180#defineHCCCL_NCF_SEND_MBX_ACKM6181#defineHCCCL_NCF_SEND_MBX_ACKM6182#defineHCCCL_NCF_PDO_OUT_ACKM6183#defineHCCCL_NCF_PDO_UT_ACKM6186#defineHCCCL_NCF_PDO_IN_CMDM6185#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6188#defineHCCCL_NCF_PDI_SCS_CMDM6188		
#defineHCCCU_USHOSTFlagsM6168#defineHCCCU_HCF_HOST_COS_CMDM6169#defineHCCCU_HCF_SEND_MBX_CMDM6171#defineHCCCU_HCF_SEND_MBX_CMDM6171#defineHCCCU_HCF_PDO_OUT_CMDM6172#defineHCCCU_HCF_PDO_IN_ACKM6173#defineHCCCU_HCF_PDI_OUT_CMDM6175#defineHCCCL_HCF_PDI_OUT_CMDM6176#defineHCCCL_USNetxFlagsM6177#defineHCCCL_NCF_COMMUNICATINGM6178#defineHCCCL_NCF_HOST_COS_ACKM6180#defineHCCCL_NCF_SEND_MBX_ACKM6181#defineHCCCL_NCF_SEND_MBX_ACKM6182#defineHCCCL_NCF_PDO_OUT_ACKM6183#defineHCCCL_NCF_PDO_UT_ACKM6186#defineHCCCL_NCF_PDO_IN_CMDM6185#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6186#defineHCCCL_NCF_PDO_IN_CMDM6188#defineHCCCL_NCF_PDI_SCS_CMDM6188	#define HCCCU_NCF_PD1_IN_CMD	
#defineHCCC0_HCF_NETX_COS_ACKM6170#defineHCCC0_HCF_SEND_MBX_CMDM6171#defineHCCC0_HCF_RECV_MBX_ACKM6172#defineHCCC0_HCF_PD0_OUT_CMDM6173#defineHCCC0_HCF_PD1_OUT_CMDM6175#defineHCCC0_HCF_PD1_IN_ACKM6176#defineHCCC1_USNetxFlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MBX_ACKM6183#defineHCCC1_NCF_PD0_OUT_ACKM6183#defineHCCC1_NCF_PD0_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6186#defineHCCC1_NCF_PD1_IN_CMDM6186#defineHCCC1_NCF_PD1_IN_CMDM6186#defineHCCC1_NCF_PD1_IN_CMDM6188#defineHCCC1_NCF_PD1_IN_CMDM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188#defineHCCC1_USFJEAgsM6188 </td <td>#define HCCCU_usHostFlags</td> <td></td>	#define HCCCU_usHostFlags	
#defineHCCC0_HCF_SEND_MBX_CMDM6171#defineHCCC0_HCF_RECV_MBX_ACKM6172#defineHCCC0_HCF_PD0_OUT_CMDM6173#defineHCCC0_HCF_PD0_IN_ACKM6174#defineHCCC0_HCF_PD1_OUT_CMDM6175#defineHCCC0_HCF_PD1_IN_ACKM6176#defineHCCC1_usNetxFlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_ERRORM6179#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_RECV_MBX_ACKM6182#defineHCCC1_NCF_RECV_MBX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_USHostFlagsM6188#defineHCCC1_USHostFlagsM6188#defineHCCC1_USHostFlagsM6188#defineHCC1_HCF_HOST_COS_CMDM6189		
#defineHCCC0_HCF_RECV_MEX_ACKM6172#defineHCCC0_HCF_PD0_OUT_CMDM6173#defineHCCC0_HCF_PD1_IN_ACKM6174#defineHCCC0_HCF_PD1_IN_ACKM6175#defineHCCC1_USNetxFlagsM6176#defineHCCC1_USNetxFlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_ERRORM6179#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_NETX_COS_CMDM6182#defineHCCC1_NCF_RECV_MEX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD1_N_CMDM6185#defineHCCC1_NCF_PD1_N_CMDM6187#defineHCCC1_NCF_PD1_N_CMDM6187#defineHCCC1_NCF_PD1_N_CMDM6187#defineHCCC1_NCF_PD1_N_CMDM6187#defineHCCC1_NCF_PD1_N_CMDM6187#defineHCCC1_NCF_PD1_N_CMDM6187#defineHCC1_NCF_PD1_N_CMDM6187#defineHCC1_NCF_PD1_N_CMDM6187#defineHCC1_NCF_PD1_N_CMDM6187#defineHCC1_NCF_PD1_N_CMDM6187#defineHCC1_NCF_PD1_N_CMDM6188#defineHCC1_NCF_PD1_N_CMDM6188#defineHCC1_HCF_HOST_COS_CMDM6189		
#defineHCCC0_HCF_PD0_OUT_CMDM6173#defineHCCC0_HCF_PD0_IN_ACKM6174#defineHCCC0_HCF_PD1_OUT_CMDM6175#defineHCCC0_HCF_PD1_IN_ACKM6176#defineHCCC1_usNetxFlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_ERRORM6179#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MBX_ACKM6182#defineHCCC1_NCF_PD0_OUT_ACKM6183#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6188#defineHCCC1_NCF_PD1_IN_CMDM6188#defineHCCC1_NCF_PD1_IN_CMDM6189		
#defineHCCC0_HCF_PD0_IN_ACKM6174#defineHCCC0_HCF_PD1_OUT_CMDM6175#defineHCCC0_HCF_PD1_IN_ACKM6176#defineHCCC1_USNetxFlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_ERRORM6179#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MEX_ACKM6182#defineHCCC1_NCF_RECV_MEX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_UT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6188#defineHCCC1_NCF_PD1_IN_CMDM6188#defineHCCC1_NCF_PD1_IN_CMDM6188#defineHCCC1_NCF_PD1_IN_CMDM6189		
#defineHCCC0_HCF_PD1_OUT_CMDM6175#defineHCCC0_HCF_PD1_IN_ACKM6176#defineHCCC1_usNetxFlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_ERRORM6179#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MBX_ACKM6182#defineHCCC1_NCF_RECV_MBX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_NCF_PD1_IN_CMDM6188#defineHCCC1_USHostFlagsM6188#defineHCCC1_USHostFlagsM6189		
#defineHCCC0_HCF_PD1_IN_ACKM6176#defineHCCC1_usNetxFlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_ERRORM6179#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MBX_ACKM6182#defineHCCC1_NCF_RECV_MBX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_USHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		
#defineHCCC1_usNetxFlagsM6177#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_ERRORM6179#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MEX_ACKM6182#defineHCCC1_NCF_RECV_MEX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_usHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		
#defineHCCC1_NCF_COMMUNICATINGM6178#defineHCCC1_NCF_ERRORM6179#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MBX_ACKM6182#defineHCCC1_NCF_RECV_MBX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_USHostFlagsM6188#defineHCCC1_USHostFlagsM6189		
#defineHCCC1_NCF_ERRORM6179#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MBX_ACKM6182#defineHCCC1_NCF_RECV_MBX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_USHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		
#defineHCCC1_NCF_HOST_COS_ACKM6180#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MBX_ACKM6182#defineHCCC1_NCF_RECV_MBX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_USHOSTFLagsM6188#defineHCCC1_USHOSTFLagsM6189		
#defineHCCC1_NCF_NETX_COS_CMDM6181#defineHCCC1_NCF_SEND_MBX_ACKM6182#defineHCCC1_NCF_RECV_MBX_CMDM6183#defineHCCC1_NCF_PDO_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_USHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		
#defineHCCC1_NCF_SEND_MBX_ACKM6182#defineHCCC1_NCF_RECV_MBX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_usHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		
#defineHCCC1_NCF_RECV_MBX_CMDM6183#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_usHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		M6181
#defineHCCC1_NCF_PD0_OUT_ACKM6184#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_usHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		M6182
#defineHCCC1_NCF_PD0_IN_CMDM6185#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_usHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		M6183
#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_usHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		M6184
#defineHCCC1_NCF_PD1_OUT_ACKM6186#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_usHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189	#define HCCC1_NCF_PD0_IN_CMD	M6185
#defineHCCC1_NCF_PD1_IN_CMDM6187#defineHCCC1_usHostFlagsM6188#defineHCCC1_HCF_HOST_COS_CMDM6189		M6186
#define HCCC1_HCF_HOST_COS_CMD M6189		M6187
		M6188
		M6189
#define HCCCI_HCF_NETX_COS_ACK M6190	#define HCCC1_HCF_NETX_COS_ACK	M6190

#define HCCC1 HCF SEND MBX CMD	M6191
	HOLDL
#define HCCC1 HCF RECV MBX ACK	M6192
#define HCCC1 HCF PD0 OUT CMD	M6193
#define HCCCI_HCF_PDU_IN_ACK	M6194
#define HCCC1 HCF PD1 OUT CMD	M6195
#dofine HCCC1 HCE DD1 TN ACK	METRE
	MOLDO
#define HCCC2_usNetxFlags	M6197
#define HCCC2 NCF COMMUNICATING	M6198
Hala fina UGOGO NOR ERROR	MC100
#deline hccc2_hcr_Error	M0199
#define HCCC2 NCF HOST COS ACK	M6200
#define HCCC2 NCF NETX COS CMD	M6201
#define HOGO2 NOT GEND MDV NOV	MCOOO
#deline HCCC2_NCF_SEND_MBX_ACK	M6202
#define HCCC2 NCF RECV MBX CMD	M6203
<pre>#define HCCC1_HCF_SEND_MBX_CMD #define HCCC1_HCF_RECV_MBX_ACK #define HCCC1_HCF_PD0_OUT_CMD #define HCCC1_HCF_PD0_IN_ACK #define HCCC1_HCF_PD1_OUT_CMD #define HCCC2_usNetxFlags #define HCCC2_usNetxFlags #define HCCC2_NCF_COMMUNICATING #define HCCC2_NCF_ERROR #define HCCC2_NCF_HOST_COS_ACK #define HCCC2_NCF_NETX_COS_CMD #define HCCC2_NCF_SEND_MBX_ACK #define HCCC2_NCF_RECV_MBX_CMD #define HCCC2_NCF_PD0_OUT_ACK #define HCCC2_NCF_PD0_IN_CMD #define HCCC2_NCF_PD1_OUT_ACK #define HCCC2_NCF_PD1_IN_CMD</pre>	M6204
	10207
#define HCCC2_NCF_PDU_IN_CMD	M6205
#define HCCC2 NCF PD1 OUT ACK	M6206
#define HCCC2 NCE PD1 IN CMD	M6206 M6207
	10207
#define HCCC2_usHostFlags	M6208
#define HCCC2 HCF HOST COS CMD	м6209
#dofing UCCC2 UCE NETY COS ACK	M6210
#deline hccc2_hcr_heix_co3_kck	M0210
#define HCCC2_HCF_SEND_MBX_CMD	MbZll
#define HCCC2 HCF RECV MBX ACK	M6212
#define HCCC2 HCE PDO OUT CMD	M6213
#define HCCC2_HCF_PD0_IN_ACK	M6214
#define HCCC2 HCF PD1 OUT CMD	M6215
#dofino HCCC2 HCE DD1 IN ACK	M6216
#deline nocc_ncr_PDI_IN_ACK	110210
#define HCCC3 usNetxFlags	M6217
#define HCCC3 NCF COMMUNICATING	M6218
	NC210
#deline HUCUS_NUF_ERROR	M6219
#define HCCC3 NCF HOST COS ACK	M6220
#define HCCC3 NCE NETX COS CMD	M6221
A define hoods Not NEW YOR YOR	
#define HCCC3_NCF_SEND_MBX_ACK	M6222
#define HCCC3 NCF RECV MBX CMD	M6223
#define HCCC3 NCE PDO OUT ACK	M6224
A define heeds Not The out Not	
#define HCCC3_NCF_PDU_IN_CMD	M6225
#define HCCC3 NCF PD1 OUT ACK	M6226
#define HCCC3 NCF PD1 IN CMD	M6227
	M0227
#define HCCC3_usHostFlags	M6228
#define HCCC3 HCF HOST COS CMD	м6229
#dofing UCCC3 UCE NETY COS ACK	ME220
#deline necco_ner_neix_cos_Ack	M0230
#define HCCC3 HCF SEND MBX CMD	M6231
#define HCCC3 HCF RECV MBX ACK	м6232
#dofine UCCC2 UCE DD0 OUT CMD	MCOOO
#deline acces_acr_ph0_001_CMD	M0255
#define HCCC3_HCF_PD0_IN_ACK	M6234
#define HCCC3 HCF PD1 OUT CMD	M6235
<pre>#define HCCC2_NCF_PD0_IN_CMD #define HCCC2_NCF_PD1_OUT_ACK #define HCCC2_NCF_PD1_IN_CMD #define HCCC2_USHOSTFlags #define HCCC2_USHOSTFLAGS #define HCCC2_HCF_NETX_COS_CMD #define HCCC2_HCF_NETX_COS_ACK #define HCCC2_HCF_RECV_MBX_ACK #define HCCC2_HCF_PD0_OUT_CMD #define HCCC2_HCF_PD0_IN_ACK #define HCCC2_HCF_PD1_IN_ACK #define HCCC2_HCF_PD1_IN_ACK #define HCCC3_NCF_COMMUNICATING #define HCCC3_NCF_ERROR #define HCCC3_NCF_ERROR #define HCCC3_NCF_NETX_COS_ACK #define HCCC3_NCF_NETX_COS_ACK #define HCCC3_NCF_NETX_COS_ACK #define HCCC3_NCF_RECV_MBX_ACK #define HCCC3_NCF_PD0_OUT_ACK #define HCCC3_NCF_PD0_OUT_ACK #define HCCC3_NCF_PD0_IN_CMD #define HCCC3_NCF_PD1_IN_CMD #define HCCC3_NCF_PD1_IN_CMD #define HCCC3_NCF_PD1_IN_CMD #define HCCC3_NCF_PD1_IN_CMD #define HCCC3_HCF_HOST_COS_ACK #define HCCC3_NCF_PD1_IN_CMD #define HCCC3_NCF_PD1_IN_CMD #define HCCC3_HCF_HOST_COS_ACK #define HCCC3_HCF_PD1_IN_CMD #define HCCC3_HCF_PD0_OUT_ACK #define HCCC3_HCF_PD1_IN_CMD #define HCCC3_HCF_SEND_MBX_CMD #define HCCC3_HCF_PD0_OUT_CMD #define HCCC3_HCF_PD0_IN_CMD #define HCCC3_HCF_PD0_IN_ACK #define HCCC3_HCF_PD0_IN_ACK #define HCCC3_HCF_PD1_OUT_CMD #define HCCC3_HCF_PD1_OUT_CMD #define HCCC3_HCF_PD1_OUT_CMD #define HCCC3_HCF_PD1_OUT_CMD #define HCCC3_HCF_PD1_OUT_CMD #define HCCC3_HCF_PD1_OUT_CMD #define HCCC3_HCF_PD1_IN_ACK #define HCCC3_HCF_PD1_IN_ACK #define HCCC3_HCF_PD1_IN_ACK #define HCCC3_HCF_PD1_IN_ACK #define HCCC3_HCF_PD1_IN_ACK #define HCCC3_HCF_PD1_IN_ACK #define HCCC3_HCF_PD1_IN_ACK #define HCCC3_HCF_PD1_IN_ACK</pre>	M6236
THE RECEIPTING ACK	10230
#define HCAC0_usNetxFlags	M6237
#define HCAC0_NCF_COMMUNICATING #define HCAC0_NCF_ERROR	M6238
"define noneo_ner_connentine	1102.50
	M6239
#define HCAC0_NCF_HOST_COS_ACK	M6240
#define HCAC0 NCF NETX COS CMD	M6241
#define HCAC0_NCF_SEND_MBX_ACK	M6242
#define HCAC0 NCF RECV MBX CMD	M6243
#define HCAC0 NCF PD0 OUT ACK	M6244
#define HCAC0_NCF_PD0_IN_CMD	M6245
#define HCAC0 NCF PD1 OUT ACK	M6246
#define HCAC0 NCF PD1 IN CMD	M6247
<pre>#define HCAC0_usHostFlags</pre>	M6248
#define HCAC0_HCF_HOST_COS_CMD	M6249
#define HCACO HCF NETX COS ACK	M6250
#define HCAC0_HCF_SEND_MBX_CMD	M6251
#define HCAC0 HCF RECV MBX ACK	M6252
#define HCAC0 HCF PD0 OUT CMD	M6253
#define HCAC0_HCF_PD0_IN_ACK	M6254
#define HCAC0 HCF PD1 OUT CMD	M6255
#define HCAC0 HCF PD1 IN ACK	M6256
#define HCAC1_usNetxFlags	M6257
#define HCAC1 NCF COMMUNICATING	M6258
#define HCAC1 NCF ERROR	M6259
#define HCAC1_NCF_HOST_COS_ACK	M6260
	MCOCI
#define HCAC1 NCF NETX COS CMD	M6261

#define HCAC1_NCF_SEND_MBX_ACK	M6262
#define HCAC1_NCF_SEND_MBX_ACK #define HCAC1_NCF_RECV_MBX_CMD #define HCAC1_NCF_PD0_OUT_ACK #define HCAC1_NCF_PD0_IN_CMD	M6263
#define HCAC1_NCF_PD0_OUT_ACK	M6264
#define HCAC1 NCF PD0 IN CMD	M6265
#define HCAC1 NCF PD1 OUT ACK	M6266
#define HCAC1 NCE PD1 IN CMD	M6267
#define HCAC1 HCF HOST COS CMD	M6269
#define HCAC1_HCF_NETX_COS_ACK	M6268 M6269 M6270
	10270
<pre>#define HCAC1_HCF_SEND_MBX_CMD #define HCAC1_HCF_RECV_MBX_ACK #define HCAC1_HCF_PD0_OUT_CMD #define HCAC1_HCF_PD0_IN_ACK</pre>	M6271
#define HCACI_HCF_RECV_MBX_ACK	M6272
#define HCAC1_HCF_PDU_OUT_CMD	M6273
#define HCAC1_HCF_PD0_IN_ACK	M6274
#define HCACI HCF PDI OUT CMD	M6275
#define HCAC1_HCF_PD1_IN_ACK	M6276 M6277
#define CC0_RCX_APP_COS_APP_READY	M6277
#define CC0 RCX APP COS BUS ON	M6278
#define CC0 RCX APP COS BUS ON ENABLE	M6279
#define CCO RCX APP COS INIT	M6280
	M6281
	M6282
#define CCO RCX APP COS LOCK OFC FNA	M6283
#define CCO_RCX_APP_COS_LOCK_CFG_ENA #define CCO_RCX_APP_COS_DMA	M6284
Hadfing COL ROA AFF_COD_DMA	
<pre>#define CC0_RCX_APP_COS_DMA_ENABLE #define CC0_ulDeviceWatchdog</pre>	M6285 M6286
#define CCU_ulDeviceWatchdog	
#define CC0 RCX COMM COS READY	M6287
#define CC0_RCX_COMM_COS_RUN	M6288
#define CC0_RCX_COMM_COS_BUS_ON	M6289
	M6290
#define CC0 RCX COMM COS CONFIG NEW	M6291
#define CC0 RCX COMM COS RESTART REQ	M6292
#define CC0 RCX COMM COS RESTART REQ ENA	M6293
#define CC0_RCX_COMM_COS_DMA	M6293 M6294
#define CCO ulCommunicationState	M6295
#define CCO_ulCommunicationError	M6296
#define CC0_usVersion	M6297
#define CCO_usWatchdogTime	M6298
#define CC0_bPDInHskMode	M6299
#define CC0_bPDInSource	M6300
#define CC0_bPDOutHskMode	M6301
#define CC0 bPDOutSource	M6302
#define CC0_ulHostWatchdog	M6303
#define CC0_ulErrorCount	M6303 M6304
#define CC0 bErrorLogInd	M6305
#define CC0 bErrorPDInCnt	M6306
#define CC0 bErrorPDOutCnt	M6307
#define CCO_bErrorSyncCnt	M6308
#define CC0_bSyncHskMode	M6309
#define CC0_bSyncSource	M6310
#define CCO_ulSlaveState	M6311
#define CC0_ulSlaveErrLogInd	M6312
#define CC0_ulNumOfConfigSlaves	M6313
#define CC0_ulNumOfActiveSlaves	M6314
#define CC0_ulNumOfDiagSlaves	M6315
#define CC1 RCX APP COS APP READY	M6316
#define CC1 RCX APP COS BUS ON	M6317
#define CC1 RCX APP COS BUS ON ENABLE	M6318
#define CC1 RCX APP COS INIT	M6319
#define CC1_RCX_APP_COS_INIT_ENABLE	M6320
#define CC1_RCX_APP_COS_LOCK_CFG	M6321
#define CC1_RCX_APP_COS_LOCK_CFG_ENA	M6322
#define CC1_RCX_APP_COS_DMA	M6323
#define CC1_RCX_APP_COS_DMA_ENABLE	M6324
#define CC1_ulDeviceWatchdog	M6325
#define CC1 RCX COMM COS READY	M6326
#define CC1 RCX COMM COS RUN	M6327
#define CC1 RCX COMM COS BUS ON	M6328
#define CC1 RCX COMM COS CONFIG LOCKED	M6329
#define CC1 RCX COMM COS CONFIG NEW	M6329 M6330
#define CC1 RCX COMM COS RESTART REQ	M6330 M6331
#define CC1_RCX_COMM_COS_RESTART_REQ #define CC1 RCX_COMM_COS_RESTART_REQ_ENA	
#GETTHE CCT KCA COMM COS KESTART KEQ ENA	M6332

#define CC1_RCX_COMM_COS_DMA	M6333
#define CC1_ulCommunicationState	M6334
#define CC1_ulCommunicationError	M6335
#define CC1 usVersion	M6336
#define CC1 usWatchdogTime	M6337
#define CC1 bPDInHskMode	M6338
#define CC1 bPDInSource	M6339
#define CC1_bPDOutHskMode	M6340
#define CC1 bPDOutSource	M6341
#define CC1_ulHostWatchdog	M6342
#define CC1_ulErrorCount	M6343
#define CC1 bErrorLogInd	M6344
#define CC1 bErrorPDInCnt	M6345
#define CC1 bErrorPDOutCnt	M6346
#define CC1 bErrorSyncCnt	M6347
#define CC1 bSyncHskMode	M6348
#define CC1 bSyncSource	M6349
#define CC1_ulSlaveState	M6350
#define CC1_ulSlaveErrLogInd	M6351
#define CC1_ulNumOfConfigSlaves	M6352
#define CC1_ulNumOfActiveSlaves	M6353
#define CC1_ulNumOfDiagSlaves	M6354

MacroNameDefinition \$6C000.pmc File Content
CLOSE
END GAT
DEL GAT
#Include "MacroNameDefinition_\$6C000.h"
SI abCookie 0 ->Y:\$6C000,0,8
SI abCookie 1 ->Y:\$6C000,8,8
SI abCookie 2 ->X:\$6C000,0,8
SI_abCookie_2_ > A:\$00000,0,0 SI abCookie_3 ->X:\$6C000,8,8
SI_ulDpmTotalSize->DP:\$6C001
SI ulDeviceNumber->DP:\$6C002
SI ulserialNumber->DP:\$6C003
SI ausHwOptions 0 ->Y:\$6C004,0,16
SI ausHwOptions 1 ->X:\$6C004,0,16
SI ausHwOptions 2 ->Y:\$6C005,0,16
SI ausHwOptions 3 ->X:\$6C005,0,16
SI usManufacturer->Y:\$6C006,0,16
SI usProductionDate->X:\$6C006,0,16
SI_ulLicenseFlags1->DP:\$6C007
SI ulLicenseFlags2->DP:\$6C008
SI_usNetxLicenseID->Y:\$6C009,0,16
SI usNetxLicenseFlags->X:\$6C009,0,16
SI usDeviceClass-Y:\$6C00A,0,16
SI bHwRevision->X:\$6C00A,0,8
SI bHwCompatibility->X:\$6C00A,8,8
SI bDevIdNumber->Y:\$6C00B,0,8
SCI bChannelType-Y:\$6C00C,0,8
SCI bSizePositionOfHandshake->X:\$6C00C,0,8
SCI bNumberOfBlocks->X:\$6C00C,8,8
SCI ulSizeOfChannel->DP:\$6C00D
SCI_usSizeOfMailbox->Y:\$6C00E,0,16
SCI usMailboxStartOffset->X:\$6C00E,0,16
HCI bChannelType->Y:\$6C010,0,8
HCI ulsizeOfChannel->DP:\$6C011
CCOI bChannelType->Y:\$6C014,0,8
CC01 bChannelId->Y:\$6C014,8,8
CC01 bSizePositionOfHandshake->X:\$6C014,0,8
CC01 bNumberOfBlocks->X:\$6C014,8,8
CC0I ulSizeOfChannel->DP:\$6C015
CC0I usCommunicationClass->Y:\$6C016,0,16
CC0I usProtocolClass->X:\$6C016,0,16
CC0I usConformanceClass->Y:\$6C017,0,16
CC11 bChannelType->Y:\$6C018,0,8
CC11 bChannelId->Y:\$6C018,8,8
CC11 bSizePositionOfHandshake->X:\$6C018,0,8

CC1I bNumberOfBlocks->X:\$6C018,8,8 CC1I ulSizeOfChannel->DP:\$6C019 CC1I usCommunicationClass->Y:\$6C01A,0,16 CC1I usProtocolClass->X:\$6C01A,0,16 CC1I usConformanceClass->Y:\$6C01B,0,16 CC2I bChannelType->Y:\$6C01C,0,8 CC2I bChannelId->Y:\$6C01C,8,8 CC2I bSizePositionOfHandshake->X:\$6C01C,0,8 CC2I bNumberOfBlocks->X:\$6C01C,8,8 CC2I ulSizeOfChannel->DP:\$6C01D CC2I usCommunicationClass->Y:\$6C01E,0,16 CC2I_usProtocolClass->X:\$6C01E,0,16 CC2I_usConformanceClass->Y:\$6C01F,0,16 CC3I bChannelType->Y:\$6C020,0,8 CC31 bChannelId->Y:\$6C020,8,8 CC3I_bSizePositionOfHandshake->X:\$6C020,0,8 CC3I bNumberOfBlocks->X:\$6C020,8,8 CC3I ulSizeOfChannel->DP:\$6C021 CC3I usCommunicationClass->Y:\$6C022,0,16 CC3I_usProtocolClass->X:\$6C022,0,16 CC3I usConformanceClass->Y:\$6C023,0,16 ACOI bChannelType->Y:\$6C024,0,8 ACOI bChannelId->Y:\$6C024,8,8 ACOI bSizePositionOfHandshake->X:\$6C024,0,8 ACOI bNumberOfBlocks->X:\$6C024,8,8 ACOI ulSizeOfChannel->DP:\$6C025 AC11 bChannelType->Y:\$6C028,0,8 AC1I bChannelId->Y:\$6C028,8,8 AC11 bSizePositionOfHandshake->X:\$6C028,0,8 AC1I bNumberOfBlocks->X:\$6C028,8,8 AC1I ulSizeOfChannel->DP:\$6C029 SCtrl ulSystemCommandCOS->DP:\$6C02E SStat ulSystemCOS->DP:\$6C030 SStat_ulSystemStatus->DP:\$6C031 SStat ulSystemError->DP:\$6C032 SStat ulBootError->DP:\$6C033 SStat ulTimeSinceStart->DP:\$6C034 SStat usCpuLoad->Y:\$6C035,0,16 SStat ulHWFeatures->DP:\$6C036 SSMB usPackagesAccepted->Y:\$6C040,0,16 SSMB_ulDest->DP:\$6C041 SSMB ulSrc->DP:\$6C042 SSMB_ulDestId->DP:\$6C043 SSMB ulSrcId->DP:\$6C044 SSMB ullen->DP:\$6C045 SSMB_ulId->DP:\$6C046 SSMB_ulState->DP:\$6C047 SSMB_ulCmd->DP:\$6C048 SSMB_ulExt->DP:\$6C049 SSMB_ulRout->DP:\$6C04A SSMB ultData0->DP:\$6C04B SSMB_ultData1->DP:\$6C04C SSMB ultData2->DP:\$6C04D SSMB_ultData3->DP:\$6C04E SSMB_ultData4->DP:\$6C04F SSMB ultData5->DP:\$6C050 SSMB_ultData6->DP:\$6C051 SSMB_ultData7->DP:\$6C052 SSMB ultData8->DP:\$6C053 SSMB ultData9->DP:\$6C054 SSMB_ultData10->DP:\$6C055 SSMB_ultData11->DP:\$6C056 SSMB_ultData12->DP:\$6C057 SSMB_ultData13->DP:\$6C058 SSMB ultData14->DP:\$6C059 SSMB_ultData15->DP:\$6C05A SSMB ultData16->DP:\$6C05B SSMB_ultData17->DP:\$6C05C SSMB_ultData18->DP:\$6C05D SSMB ultData19->DP:\$6C05E SSMB_ultData20->DP:\$6C05F

SRMB usWaitingPackages->Y:\$6C060,0,16 SRMB_ulDest->DP:\$6C061 SRMB_ulSrc->DP:\$6C062 SRMB_ulDestId->DP:\$6C063 SRMB ulSrcId->DP:\$6C064 SRMB_ullen->DP:\$6C065 SRMB ulId->DP:\$6C066 SRMB ulState->DP:\$6C067 SRMB_ulCmd->DP:\$6C068 SRMB_ulExt->DP:\$6C069 SRMB ulRout->DP:\$6C06A SRMB_ultData0->DP:\$6C06B SRMB_ultData1->DP:\$6C06C SRMB ultData2->DP:\$6C06D SRMB_ultData3->DP:\$6C06E SRMB ultData4->DP:\$6C06F SRMB_ultData5->DP:\$6C070 SRMB ultData6->DP:\$6C071 SRMB ultData7->DP:\$6C072 SRMB ultData8->DP:\$6C073 SRMB ultData9->DP:\$6C074 SRMB ultData10->DP:\$6C075 SRMB_ultData11->DP:\$6C076 SRMB ultData12->DP:\$6C077 SRMB ultData13->DP:\$6C078 SRMB ultData14->DP:\$6C079 SRMB ultData15->DP:\$6C07A SRMB ultData16->DP:\$6C07B SRMB_ultData17->DP:\$6C07C SRMB ultData18->DP:\$6C07D SRMB ultData19->DP:\$6C07E SRMB_ultData20->DP:\$6C07F HCSC bNetxFlags->X:\$6C080,0,8 HCSC_NSF_READY->X:\$6C080,0,1 HCSC_NSF_ERROR->X:\$6C080,1,1 HCSC_NSF_HOST_COS_ACK->X:\$6C080,2,1 HCSC NSF NETX COS CMD->X:\$6C080,3,1 HCSC_NSF_SEND_MBX_ACK->X:\$6C080,4,1 HCSC NSF RECV MBX CMD->X:\$6C080,5,1 HCSC bHostFlags->X:\$6C080,8,8 HCSC_HSF_RESET->X:\$6C080,8,1 HCSC HSF BOOTSTART->X:\$6C080,9,1 HCSC HSF HOST COS_CMD->X:\$6C080,10,1 HCSC HSF NETX COS ACK->X:\$6C080,11,1 HCSC_HSF_SEND_MBX_CMD->X:\$6C080,12,1 HCSC_HSF_RECV_MBX_ACK->X:\$6C080,13,1 HCCCO usNetxFlags->Y:\$6C082,0,16 HCCC0_NCF_COMMUNICATING->Y:\$6C082,0,1 HCCC0 NCF ERROR->Y:\$6C082,1,1 HCCC0 NCF HOST COS_ACK->Y:\$6C082,2,1 HCCC0 NCF NETX COS CMD->Y:\$6C082,3,1 HCCC0 NCF SEND MBX ACK->Y:\$6C082,4,1 HCCC0 NCF RECV MBX CMD->Y:\$6C082,5,1 HCCCO NCF PD0 OUT ACK->Y:\$6C082,6,1 HCCC0_NCF_PD0_IN_CMD->Y:\$6C082,7,1 HCCC0_NCF_PD1_OUT_ACK->Y:\$6C082,8,1 HCCC0_NCF_PD1_IN_CMD->Y:\$6C082,9,1 HCCC0_usHostFlags->X:\$6C082,0,16 HCCC0_HCF_HOST_COS_CMD->X:\$6C082,2,1 HCCC0_HCF_NETX_COS_ACK->X:\$6C082,3,1 HCCC0 HCF SEND MBX CMD->X:\$6C082,4,1 HCCC0_HCF_RECV_MBX_ACK->X:\$6C082,5,1 HCCC0_HCF_PD0_OUT_CMD->X:\$6C082,6,1 HCCC0_HCF_PD0_IN_ACK->X:\$6C082,7,1 HCCC0 HCF PD1 OUT CMD->X:\$6C082,8,1 HCCC0_HCF_PD1_IN_ACK->X:\$6C082,9,1 HCCC1_usNetxFlags->Y:\$6C083,0,16 HCCC1 NCF COMMUNICATING->Y:\$6C083,0,1 HCCC1_NCF_ERROR->Y:\$6C083,1,1 HCCC1_NCF_HOST_COS_ACK->Y:\$6C083,2,1 HCCC1_NCF_NETX_COS_CMD->Y:\$6C083,3,1

HCCC1 NCF SEND MBX ACK->Y:\$6C083,4,1 HCCC1_NCF_RECV_MBX_CMD->Y:\$6C083,5,1 HCCC1 NCF PD0 OUT ACK->Y:\$6C083,6,1 HCCC1_NCF_PD0_IN_CMD->Y:\$6C083,7,1 HCCC1 NCF PD1 OUT ACK->Y:\$6C083,8,1 HCCC1_NCF_PD1_IN_CMD->Y:\$6C083,9,1 HCCC1 usHostFlags->X:\$6C083,0,16 HCCC1 HCF HOST COS CMD->X:\$6C083,2,1 HCCC1_HCF_NETX_COS_ACK->X:\$6C083,3,1 HCCC1_HCF_SEND_MBX_CMD->X:\$6C083,4,1 HCCC1_HCF_RECV_MBX_ACK->X:\$6C083,5,1 HCCC1 HCF PD0 OUT CMD->X:\$6C083,6,1 HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 HCCC1 HCF PD1 IN ACK->X:\$6C083,9,1 HCCC2_usNetxFlags->Y:\$6C084,0,16 HCCC2 NCF COMMUNICATING->Y:\$6C084,0,1 HCCC2 NCF ERROR->Y:\$6C084,1,1 HCCC2 NCF HOST COS ACK->Y:\$6C084,2,1 HCCC2_NCF_NETX_COS_CMD->Y:\$6C084,3,1 HCCC2_NCF_SEND_MBX_ACK->Y:\$6C084,4,1 HCCC2 NCF RECV MBX CMD->Y:\$6C084,5,1 HCCC2 NCF PD0 OUT ACK->Y:\$6C084,6,1 HCCC2_NCF_PD0_IN_CMD->Y:\$6C084,7,1 HCCC2_NCF_PD1_OUT_ACK->Y:\$6C084,8,1 HCCC2 NCF PD1 IN CMD->Y:\$6C084,9,1 HCCC2_usHostFlags->X:\$6C084,0,16 HCCC2 HCF HOST COS CMD->X:\$6C084,2,1 HCCC2 HCF NETX COS ACK->X:\$6C084,3,1 HCCC2_HCF_SEND_MBX_CMD->X:\$6C084,4,1 HCCC2_HCF_RECV_MBX_ACK->X:\$6C084,5,1 HCCC2_HCF_PD0_OUT_CMD->X:\$6C084,6,1 HCCC2 HCF PD0 IN ACK->X:\$6C084,7,1 HCCC2_HCF_PD1_OUT_CMD->X:\$6C084,8,1 HCCC2_HCF_PD1_IN_ACK->X:\$6C084,9,1 HCCC3 usNetxFlags->Y:\$6C085,0,16 HCCC3 NCF_COMMUNICATING->Y:\$6C085,0,1 HCCC3_NCF_ERROR->Y:\$6C085,1,1 HCCC3_NCF_HOST_COS_ACK->Y:\$6C085,2,1 HCCC3 NCF NETX COS CMD->Y:\$6C085,3,1 HCCC3 NCF SEND MBX ACK->Y:\$6C085,4,1 HCCC3 NCF RECV MBX CMD->Y:\$6C085,5,1 HCCC3_NCF_PD0_OUT_ACK->Y:\$6C085,6,1 HCCC3 NCF PD0 IN CMD->Y:\$6C085,7,1 HCCC3_NCF_PD1_OUT_ACK->Y:\$6C085,8,1 HCCC3_NCF_PD1_IN_CMD->Y:\$6C085,9,1 HCCC3 usHostFlags->X:\$6C085,0,16 HCCC3_HCF_HOST_COS_CMD->X:\$6C085,2,1 HCCC3 HCF NETX COS ACK->X:\$6C085,3,1 HCCC3 HCF SEND MBX CMD->X:\$6C085,4,1 HCCC3 HCF RECV MBX ACK->X:\$6C085,5,1 HCCC3_HCF_PD0_OUT_CMD->X:\$6C085,6,1 HCCC3_HCF_PD0_IN_ACK->X:\$6C085,7,1 HCCC3 HCF PD1 OUT CMD->X:\$6C085,8,1 HCCC3_HCF_PD1_IN_ACK->X:\$6C085,9,1 HCAC0 usNetxFlags->Y:\$6C086,0,16 HCAC0 NCF COMMUNICATING->Y:\$6C086,0,1 HCAC0 NCF ERROR->Y:\$6C086,1,1 HCAC0_NCF_HOST_COS_ACK->Y:\$6C086,2,1 HCAC0_NCF_NETX_COS_CMD->Y:\$6C086,3,1 HCACO NCF SEND MBX ACK->Y:\$6C086,4,1 HCAC0_NCF_RECV_MBX_CMD->Y:\$6C086,5,1 HCACO NCF PD0 OUT ACK->Y:\$6C086,6,1 HCACO NCF PDO IN CMD->Y:\$6C086,7,1 HCAC0 NCF PD1 OUT ACK->Y:\$6C086,8,1 HCAC0_NCF_PD1_IN_CMD->Y:\$6C086,9,1 HCAC0 usHostFlags->X:\$6C086,0,16 HCACO HCF HOST COS CMD->X:\$6C086,2,1 HCAC0_HCF_NETX_COS_ACK->X:\$6C086,3,1 HCAC0 HCF SEND MBX CMD->X:\$6C086,4,1 HCAC0 HCF RECV MBX ACK->X:\$6C086,5,1

HCAC0 HCF PD0 OUT CMD->X:\$6C086,6,1 HCAC0_HCF_PD0_IN_ACK->X:\$6C086,7,1 HCAC0_HCF_PD1_OUT_CMD->X:\$6C086,8,1 HCAC0_HCF_PD1_IN_ACK->X:\$6C086,9,1 HCAC1 usNetxFlags->Y:\$6C087,0,16 HCAC1_NCF_COMMUNICATING->Y:\$6C087,0,1 HCAC1 NCF ERROR->Y:\$6C087,1,1 HCAC1 NCF HOST COS ACK->Y:\$6C087,2,1 HCAC1 NCF NETX COS CMD->Y:\$6C087,3,1 HCAC1_NCF_SEND_MBX_ACK->Y:\$6C087,4,1 HCAC1_NCF_RECV_MBX_CMD->Y:\$6C087,5,1 HCAC1 NCF PD0 OUT ACK->Y:\$6C087,6,1 HCAC1_NCF_PD0_IN_CMD->Y:\$6C087,7,1 HCAC1_NCF_PD1_OUT_ACK->Y:\$6C087,8,1 HCAC1 NCF PD1 IN CMD->Y:\$6C087,9,1 HCAC1 usHostFlags->X:\$6C087,0,16 HCAC1_HCF_HOST_COS_CMD->X:\$6C087,2,1 HCAC1_HCF_NETX_COS_ACK->X:\$6C087,3,1 HCAC1 HCF SEND MBX CMD->X:\$6C087,4,1 HCAC1_HCF_RECV_MBX_ACK->X:\$6C087,5,1 HCAC1_HCF_PD0_OUT_CMD->X:\$6C087,6,1 HCAC1_HCF_PD0_IN_ACK->X:\$6C087,7,1 HCAC1_HCF_PD1_OUT_CMD->X:\$6C087,8,1 HCAC1_HCF_PD1_IN_ACK->X:\$6C087,9,1 CCO RCX APP COS APP READY->Y:\$6C0C2,0,1 CCO RCX APP COS BUS ON->Y:\$6C0C2,1,1 CC0_RCX_APP_COS_BUS_ON_ENABLE->Y:\$6C0C2,2,1 CCO RCX APP COS INIT->Y:\$6C0C2,3,1 CCO RCX APP COS INIT ENABLE->Y:\$6C0C2,4,1 CC0_RCX_APP_COS_LOCK_CFG->Y:\$6C0C2,5,1 CCO RCX APP COS LOCK CFG ENA->Y:\$6C0C2,6,1 CCO RCX APP COS DMA->Y:\$6C0C2,7,1 CCO RCX APP COS DMA ENABLE->Y:\$6C0C2,8,1 CC0_ulDeviceWatchdog->DP:\$6C0C3 CCO RCX COMM COS READY->Y:\$6C0C4,0,1 CC0 RCX COMM COS RUN->Y:\$6C0C4,1,1 CC0_RCX_COMM_COS_BUS_ON->Y:\$6C0C4,2,1 CC0_RCX_COMM_COS_CONFIG_LOCKED->Y:\$6C0C4,3,1 CC0_RCX_COMM_COS_CONFIG_NEW->Y:\$6C0C4,4,1 CC0 RCX COMM COS RESTART REQ->Y:\$6C0C4,5,1 CC0_RCX_COMM_COS_RESTART_REQ_ENA->Y:\$6C0C4,6,1 CC0 RCX COMM COS DMA->Y:\$6C0C4,7,1 CC0 ulCommunicationState->DP:\$6C0C5 CC0 ulCommunicationError->DP:\$6C0C6 CCO usVersion->Y:\$6C0C7,0,16 CC0 usWatchdogTime->X:\$6C0C7,0,16 CC0 bPDInHskMode->Y:\$6C0C8,0,8 CC0 bPDInSource->Y:\$6C0C8,8,8 CC0 bPDOutHskMode->X:\$6C0C8,0,8 CC0 bPDOutSource->X:\$6C0C8,8,8 CC0 ulHostWatchdog->DP:\$6C0C9 CC0_ulErrorCount->DP:\$6C0CA CC0 bErrorLogInd->Y:\$6C0CB,0,8 CC0 bErrorPDInCnt->Y:\$6C0CB,8,8 CC0_bErrorPDOutCnt->X:\$6C0CB,0,8 CC0 bErrorSyncCnt->X:\$6C0CB,8,8 CC0 bSyncHskMode->Y:\$6C0CC,0,8 CC0 bSyncSource->Y:\$6C0CC,8,8 CC0 ulSlaveState->DP:\$6C0CE CC0 ulSlaveErrLogInd->DP:\$6C0CF CC0_ulNumOfConfigSlaves->DP:\$6C0D0 CC0 ulNumOfActiveSlaves->DP:\$6C0D1 CC0_ulNumOfDiagSlaves->DP:\$6C0D2 CC1 RCX APP COS APP READY->Y:\$6D002,0,1 CC1 RCX APP COS BUS ON->Y:\$6D002,1,1 CC1 RCX APP COS BUS ON ENABLE->Y:\$6D002,2,1 CC1 RCX APP COS INIT->Y:\$6D002,3,1 CC1 RCX APP COS INIT ENABLE->Y:\$6D002,4,1 CC1_RCX_APP_COS_LOCK_CFG->Y:\$6D002,5,1 CC1 RCX APP COS LOCK CFG ENA->Y:\$6D002,6,1 CC1 RCX APP COS DMA->Y:\$6D002,7,1

CC1 RCX APP COS DMA ENABLE->Y:\$6D002,8,1
CC1 ulDeviceWatchdog->DP:\$6D003
CC1 RCX COMM COS READY->Y:\$6D004,0,1
CC1 RCX COMM COS RUN->Y:\$6D004,1,1
CC1 RCX COMM COS BUS ON->Y:\$6D004,2,1
CC1 RCX COMM COS CONFIG LOCKED->Y:\$6D004,3,1
CC1 RCX COMM COS CONFIG NEW->Y:\$6D004,4,1
CC1 RCX COMM COS RESTART REQ->Y:\$6D004,5,1
CC1 RCX COMM COS RESTART REQ ENA->Y:\$6D004,6,1
CC1 RCX COMM COS DMA->Y:\$6D004,7,1
CC1_ulCommunicationState->DP:\$6D005
CC1_ulCommunicationError->DP:\$6D006
CC1_usVersion->Y:\$6D007,0,16
CC1_usWatchdogTime->X:\$6D007,0,16
CC1_bPDInHskMode->Y:\$6D008,0,8
CC1_bPDInSource->Y:\$6D008,8,8
CC1_bPDOutHskMode->X:\$6D008,0,8
CC1_bPDOutSource->X:\$6D008,8,8
CC1_ulHostWatchdog->DP:\$6D009
CC1_ulErrorCount->DP:\$6D00A
CC1_bErrorLogInd->Y:\$6D00B,0,8
CC1_bErrorPDInCnt->Y:\$6D00B,8,8
CC1_bErrorPDOutCnt->X:\$6D00B,0,8
CC1_bErrorSyncCnt->X:\$6D00B,8,8
CC1_bSyncHskMode->Y:\$6D00C,0,8
CC1_bSyncSource->Y:\$6D00C,8,8
CC1_ulSlaveState->DP:\$6D00E
CC1_ulSlaveErrLogInd->DP:\$6D00F
CC1_ulNumOfConfigSlaves->DP:\$6D010
CC1_ulNumOfActiveSlaves->DP:\$6D011
CC1_ulNumOfDiagSlaves->DP:\$6D012

DPRAM DATA PROCESSING

Since there are two processors (i.e. UMAC and netX) attempting to access data registers in Dual-Ported Memory simultaneously, several handshaking modes can be used to guarantee data consistency. Each sub-block defines the type of handshaking, if any, it requires. The ACC-72EX Setup Assistant software output file lists the type of handshaking required for each of the sub-blocks available on the COMX module.

Should handshaking not be used, collision circuitry on the gateway will, in the very least, guarantee consistency within single byte boundaries.

Non-Cyclic Data Exchange

The mailbox of a communication channel or system channel has two areas that are used for non-cyclic message transfer to and from the netX.

Send Mailbox (System / Communication Channel)

Packet transfer from UMAC to netX firmware

Receive Mailbox (System / Communication Channel)

Packet transfer from netX firmware to UMAC

For a communication channel, send and receive mailbox areas are used by fieldbus protocols, providing a non-cyclic data exchange mechanism. Another use of the mailbox system is to allow access to the firmware running on the netX chip for diagnostic and identification purposes. The **send mailbox** is used to transfer cyclic data **to** the network or **to** the netX. The **receive mailbox** is used to transfer cyclic data **from** the netX. Modbus Plus or Ethernet TCP/IP is an example of a fieldbus protocol which utilizes a non-cyclic data exchange.

Whether or not a mailbox is used depends on the function of the firmware.



Each mailbox can hold one packet at a time. netX stores packets in an internal packet queue; these packets are not retrieved by UMAC. This queue has limited space and may fill up, so new packets may be lost. To avoid these deadlock situations, it is strongly recommended to empty the mailbox frequently, even if packets are not expected by the UMAC program. Unexpected command packets should be returned to the sender with an Unknown Command in the status field; unexpected reply messages can be discarded.

Message or Packets

The non-cyclic packets obtained through the netX mailbox have the following structure:

	Hilscher Documentation	ACC-72EX Setup Assistant
	usPackagesAccepted	SSMB_usPackagesAccepted
×	ulDest	SSMB_ulDest
Send Mailbox	ulSrc	SSMB_ulSrc
Лаіl	ulDestId	SSMB_ulDestId
⊿ pi	ulSrcId	SSMB_ulSrcId
Sen	ulLen	SSMB_ulLen
Block	ulid	SSMB_ulld
Blc	ulState	SSMB_ulState
System	ulCmd	SSMB_ulCmd
yst	ulExt	SSMB_ulExt
s	ulRout	SSMB_ulRout
		SSMB_ultData0 SSMB_ultData20

	Hilscher Documentation	ACC-72EX Setup Assistant
хо	usWaitingPackages	SRMB_usWaitingPackages
	ulDest	SRMB_ulDest
Mailbox	ulSrc	SRMB_ulSrc
	ulDestId	SRMB_ulDestId
Receive	ulSrcId	SRMB_ulSrcId
ece	ulLen	SRMB_ulLen
	ulld	SRMB_ulld
loc	ulState	SRMB_ulState
пВ	ulCmd	SRMB_ulCmd
System Block	ulExt	SRMB_ulExt
Sys	ulRout	SRMB_ulRout
		SRMB_ultData0 SRMB_ultData20

The size of a packet is always at least 40 bytes. Depending on the command, a packet may or may not have a payload in the data field (tData). If present, the contents of the data field are specific to the command or reply.

Destination Queue Handler

The ulDest field identifies a task queue in the context of the netX firmware. The task queue represents the final receiver of the packet and is assigned to a protocol stack. The ulDest field has to be filled out in any case; otherwise, the netX operating system cannot route the packet.

Source Queue Handler

The ulSrc field identifies the sender of the packet. In the context of the netX firmware (inter-task communication), this field holds the identifier of the sending task. Usually, a UMAC program uses this field for its own handle, but it can hold any handle of the sending process. The receiving task does not evaluate this field and will pass it back unchanged to the originator of the packet.

Destination Identifier

The ulDestId field identifies the destination of an unsolicited packet from the netX firmware to the UMAC. It can hold any handle that helps identify the receiver. Its use is mandatory for unsolicited packets. The receiver of unsolicited packets has to register for this service (details are yet to be determined).

Source Identifier

The ulSrcId field identifies the originator of a packet. This field is used by a UMAC program which passes a packet from an external process to an internal netX task. The ulSrcId field holds the handle of the external process. When the netX operating system returns the packet, the UMAC program can identify the packet, and returns it to the originating process. The receiving task on the netX does not evaluate this field, and passes it back unchanged. For inter-task communication, this field is not used.

Length of Data Field

The ulLen field holds the size of the data field tData in bytes. It defines the total size of the packet's payload that follows the packet's header. Note that the size of the header is not included in ulLen. Depending on the command or reply, a data field may or may not be present in a packet. If no data field is used, the length field is set to zero.

Identifier

The ulld field is used to identify a specific packet among others of the same kind. That way the application or driver can match a specific reply or confirmation packet to a previous request packet. The receiving task does not change this field and passes it back to the originator of the packet. Its use is optional in most of cases, but it is mandatory for fragmented packets. Example: downloading large amounts of data that do not fit into a single packet. For fragmented packets, the identifier field is incremented by one for every new packet.

• Status / Error Code

The ulSta field is used in response or confirmation packets. It informs the originator of the packet about success or failure of the execution of the command. The field may be also used to hold status information in a request packet. Status and error codes that may be returned in ulSta are outlined in Status and Error Code section.

Command / Response

The ulCmd field holds the command code or the response code. The command/response is specific to the receiving task. If a task is not able to execute certain commands, it will return the packet with an error indication. A command is always even (the least significant bit is zero). In the response packet, the command code is incremented by one indicating a confirmation to the request packet.

Extension

The extension field ulExt is used for controlling packets that are sent in a sequenced or fragmented manner. The extension field indicates the first, last, or a packet of a sequence. If fragmentation of packets is not required, the extension field is set to zero.

Routing Information

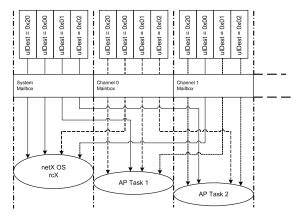
The ulRout field is used internally by the netX firmware only. It has no meaning to a driver type application and therefore is set to zero.

User Data Field

The tData field contains the payload of the packet. Depending on the command or reply, a packet may or may not have a data field. The length of the data field is given in the ulLen field.

About System and Channel Mailbox

The preferred way to address the netX operating system, rcX, is through the system mailbox. The preferred way to address a protocol stack is through its channel mailbox. All mailboxes, however, have a mechanism to route packets to any communication channel or the system channel. Therefore, the destination identifier ulDest in a packet header has to be filled in according to the targeted receiver (see the following image).



The above figure and table below illustrate the use of the destination identifier ulDest.

Value	Definition / Description
\$0	Packet is passed to the netX operating system rcX
\$1	Packet is passed to communication channel 0
\$2	Packet is passed to communication channel 1
\$3	Packet is passed to communication channel 2
\$4	Packet is passed to communication channel 3
\$20	Packet is passed to 'local' communication or system channel
Else	Reserved, Do Not Use

In regards to the channel identifier 0x00000020 (= Channel Token), the Channel Token is valid for any mailbox. That way, the UMAC program uses the same identifier for all packets without actually knowing which mailbox or communication channel is applied. The packet stays "local." The system mailbox is a little bit different because it is used to communicate to the netX operating system, rcX. The rcX has its own range of valid command codes and differs from the communication channels.

If there is a reply packet, the netX operating system returns it to the same mailbox that the request packet went through. Consequently, the UMAC program has to return its reply packet to the mailbox from which the request was received.

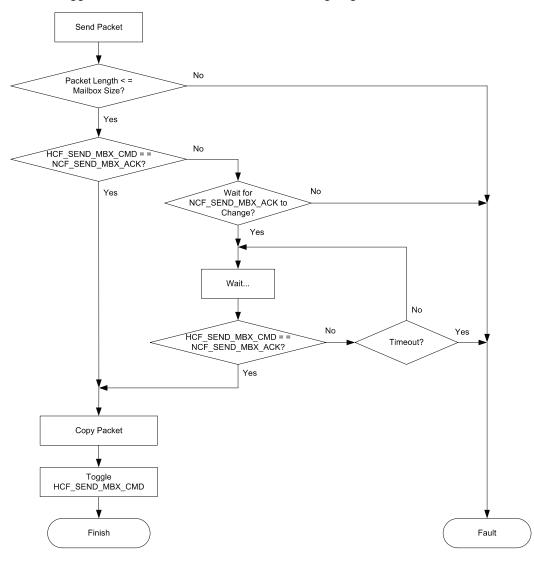
Command and Acknowledge

To ensure data consistency over the content of a mailbox, the firmware uses a pair of flags, each for one direction. Engaging these flags gives access rights alternating to either the user application or the netX firmware. If both UMAC and netX firmware were to access the mailbox at the same time, it may cause loss of data or inconsistency.

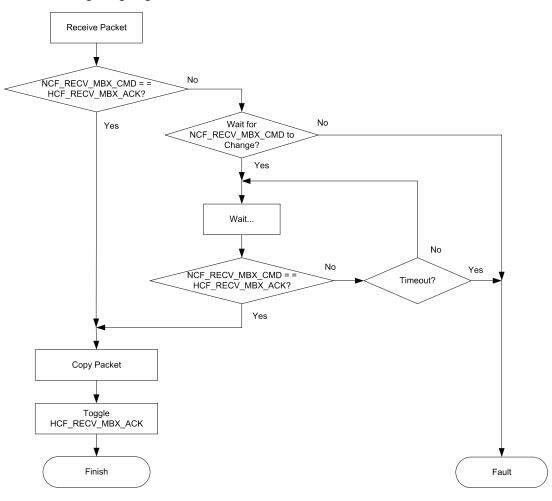
As a general rule, if both flags have the same value (both are set or both are cleared), the process which intends to write has access rights. If they have a different value, the process which intends to read has access rights. The following table illustrates this mechanism.

Send Mailbox	CMD Flag	ACK Flag	
UMAC Has Write Access	0	0	netX Has NO Read Access
UMAC Has NO Write Access	0	1	netX Has Read Access
UMAC Has NO Write Access	1	0	netX Has Read Access
UMAC Has Write Access	1	1	netX Has NO Read Access
Receive Mailbox	CMD Flag	ACK Flag	
UMAC Has NO Read Access	0	0	netX Has Write Access
UMAC Has Read Access	0	1	netX Has NO Write Access
UMAC Has Read Access	1	0	netX Has NO Write Access
UMAC Has NO Read Access	1	1	netX Has Write Access

The following flowcharts illustrate how the transfer mechanism (send and receive packets) works. In order to send a packet, first the function checks if the size of the packet to be sent exceeds the mailbox size. If both the Host Send Mailbox Command flag and the netX Send Mailbox Acknowledge flag are either set or cleared, the host application is allowed to send the packet. When copying data to the mailbox is done, the host toggles the Host Send Mailbox Command flag to give control to the netX firmware.

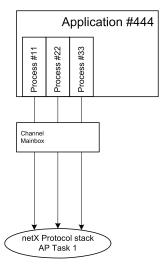


In order to receive a packet, the function checks if the netX Receive Mailbox Command flag and the Host Receive Mailbox Acknowledge flag have different values. If so, the host application is allowed to access the mailbox. When the host is done copying data from the mailbox, the host toggles the Host Receive Mailbox Acknowledge flag to give control to the netX firmware.



Using ulSrc and ulSrcId

Generally, a netX protocol stack is addressed through its communication channel mailbox. The example below shows how a host application addresses a protocol stack running in the context of the netX chip. The application is identified by a number (#444 in this example). The application consists of three processes numbered #11, #22 and #33. These processes communicate through the channel mailbox to the AP task of a protocol stack. See the following image:



Example:

This example applies to command messages imitated by a process in the context of the host application identified by number #444. If the process #22 sends a packet through the channel mailbox to the AP task, the packet header has to be filled in as follows:

Destination Queue Handler	ulDest =	32; /*	0x20: local channel mailbox */
Source Queue Handler	ulSrc =	444; /*	host application */
Destination Identifier	ulDestId=	0; /*	not used */
Source Identifier	ulSrcId =	22; /*	process number */

For packets through the channel mailbox, the application uses 32 (= 0x20, Channel Token) for the destination queue handler ulDest. The source queue handler ulSrc and the source identifier ulSrcId are used to identify the originator of a packet. The destination identifier ulDestId can be used to address certain resources in the protocol stack. It is not used in this example. The source queue handler ulSrc must have an entry, and therefore its use is mandatory; the use of ulSrcId is optional.

The netX operating system passes the request packet to the protocol stack's AP task. The protocol stack then builds a reply to the packet and returns it to the mailbox. The application has to make sure that the packet finds its way back to the originator (process #22 in the example).

How to Route rcX Packets

To route an rcX packet, the source identifier ulSrcId and the source queues handler ulSrc in the packet header hold the identification of the originating process. The router saves the original handle from ulSrcId and ulSrc. It uses handles of its own choice for ulSrcId and ulSrc before it sends the packet to the receiving process. That way, the router can identify the corresponding reply packet and match the handle from that packet with the one stored earlier. Lastly, the router replaces its handles with the original handles and returns the packet to the originating process.

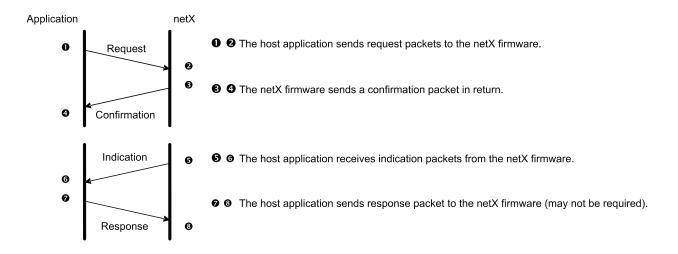
Client/Server Mechanism

Depending on the message destination or packet protocol, the UMAC program or application can act as a client or a server. This section explains both methods, but selection of the appropriate method depends on the destination or protocol option.

Application as Client

The host application may send request packets to the netX firmware at any time (transition $1 \Rightarrow 2$). Depending on the protocol stack running on the netX, parallel packets are not permitted (see protocol specific manual for details). The netX firmware sends a confirmation packet in return, signaling success or failure (transition $3 \Rightarrow 4$) while processing the request.

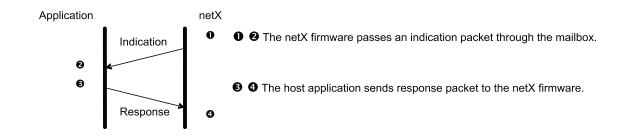
The host application has to register with the netX firmware in order to receive indication packets (transition $5 \Rightarrow 6$). Depending on the protocol stack, this is done either implicitly (if application opens a TCP/UDP socket) or explicitly (if application wants to receive unsolicited DPV1 packets). Details on when and how to register for certain events is described in the protocol specific manual. Depending on the command code of the indication packet, a response packet to the netX firmware may or may not be required (transition $7 \Rightarrow 8$).



Application as Server

The host application has to register with the netX firmware in order to receive indication packets (unsolicited telegrams). Depending on the protocol stack, this is done either implicitly (if the application opens a TCP/UDP socket) or explicitly (if the application wants to receive unsolicited DPV1 packets). Details on when and how to register for certain events is described in the protocol-specific manual.

When an appropriate event occurs and the host application is registered to receive such a notification, the netX firmware passes an indication packet through the mailbox (transition $1 \Rightarrow 2$). The host application is expected to send a response packet back to the netX firmware (transition $3 \Rightarrow 4$).



Input/Output Data Image

Hilscher products support two methods for accessing the Input/Output Data Image:

- DPM (Dual-Ported Memory) Mode
- DMA (Direct Memory Access) Mode

However, the modules used in ACC-72EX only support the DPM mode, and only Hilscher PCI cards support DMA mode.

In DPM Mode, handshaking between the UMAC (host) program and netX is required for any data transfer.

Process Data Handshake Modes

The netX firmware allows controlling the transfer of data independently for inputs and outputs. Therefore, the process data handshake is carried out individually for input and output image. The handshake cells are located in the handshake channel.

Mode	Controlled by	Consistency	Supported by	
Buffered	Host (Application/Driver)	Yes	Master & Slave Firmware	

Buffered, Host Controlled Mode

The Buffered data transfer mode can be used for both master- and slave- type devices. In "buffered" mode, the protocol stack handles the exchange of data between internal buffers and the process data images in the dual-port memory with the application via a handshake mechanism. Once copied from/into the input/output area, the host application gives control over the dual-port memory to the protocol stack. Control is given back to the host application when the protocol stack has finished copying, and so on.

The network cycle and the task cycle of the host application are not synchronized, but are consistent.



Note

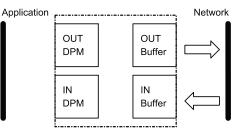
If the host application is faster than the network cycle, it might be possible that data in the output buffers is overwritten without ever being sent to the network. As for the other direction, the host application may read the same input values over several read cycles.

If the host application is slower than the network cycle, the protocol stack overwrites the input buffer with new data received from the network, which were never received by the host application. The output data on the network will be the same over several network cycles.

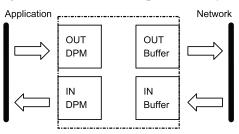
For each valid bus cycle, the protocol stack updates the process data in the internal input buffer. When the application toggles the appropriate input handshake bit, the protocol stack copies the data from the internal IN buffer into the input data image of the dual-port memory. Now the application can copy data from the dual-port memory and then give control back to the protocol stack by toggling the appropriate input handshake bit. When the application/driver toggles the output handshake bit, the protocol stack copies the data from the output data image of the dual-port memory into the internal buffer. From there, the data is transferred to the network. The protocol stack toggles the appropriate handshake bits back, indicating to the application that the transfer is finished and a new data exchange cycle may start. This mode guarantees data consistency over both the input and output areas.

Step-by-Step Procedure

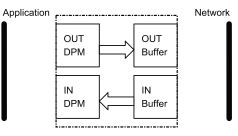
Step 1 The protocol stack sends data from the internal OUT buffer to the network and receives data from the network in the internal IN buffer.



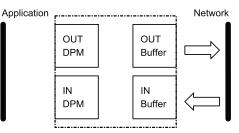
Step 2 The application has control over the dual-port memory and exchanges data with the input and output data images in the dual-port memory. The application then toggles the handshake bits, giving control over the dual-port memory to the protocol stack



Step 3 The protocol stack copies the content of the output data image into the internal OUT buffer, and from the IN buffer to the input data image.



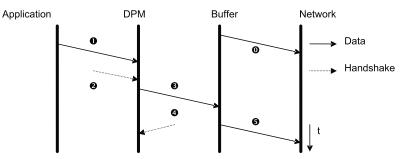
Step 4 The protocol stack toggles the handshake bits, giving control back to the application. Now, the protocol stack uses the new output data image from the OUT buffer to send it to the network, and receives data into the internal IN buffer, and then the cycle repeats.



Time-Related View

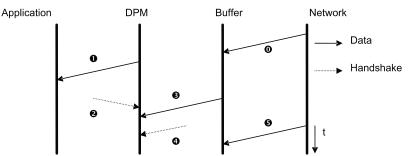
The following figure shows the procedure in a time-related view.

Output Data Exchange



- 1. The protocol stack constantly transmits data from the buffer to the network.
- 2. The application has control over the dual-port memory and can copy data to the output data image.
- 3. The application then toggles the handshake bits, giving control over the dual-port memory to the protocol stack.
- 4. The protocol stack copies the content of the output data image into the internal OUT buffer.
- 5. The protocol stack toggles the handshake bits, giving control back to the application.
- 6. Once updated, the protocol stack uses the new data from the internal buffer and sends it to the network. The cycle repeats with step 1.

Input Data Exchange



- 1. The protocol stack constantly receives data from the network into the buffer.
- 2. The application has control over the dual-port memory input data image and exchanges data with the input data image in the dual-port memory.
- 3. The application then toggles the handshake bits, giving control over the dual-port memory to the netX protocol stack.
- 4. The protocol stack copies the latest content of the internal IN buffer to the input data image of the dual-port memory.
- 5. The protocol stack then toggles the handshake bits, giving control back to the application.
- 1. The protocol stack receives data from the network into the buffer (i.e. the cycle starts over with the first step).



In case of a network fault (e.g. disconnected network cable), a slave firmware keeps the last state of the input data image. As soon as the firmware detects the network fault, it clears the Communicating flag in the netX communication flags. The input data should then no longer be evaluated.

Start / Stop Communication

Controlled or Automatic Start

The firmware has the option to start network communication after power up automatically. Whether or not the network communication will be started automatically is configurable. However, the preferred option is called "Controlled Start of Communication." This option forces the channel firmware to wait for the host application to allow network connection being opened by setting the Bus On flag in the Application Change of State register in the channel's control block. Consequently, the protocol stack will not allow the opening of network connections and does not exchange any cyclic process data until the Bus On flag is set.

The second option enables the channel firmware to open network connections automatically without interacting with the host application. It is called "Automatic Start of Communication." This method is not recommended because the host application has no control over the network connection status. In this case, the Bus On flag is not evaluated.



The Controlled Start of communication is the default method used for the default dual-port memory layout.

Start / Stop Communication through Dual-Port Memory

(Re-)Start Communication

To allow the protocol stack to open connections or to allow connections to be opened, the application sets the Bus On flag in the Application Change of State register in the channel's control block. When firmware has established a cyclic connection to at least one network mode, the channel firmware sets the Communicating flag in the netX Communication Flags register.

Stop Communication

To force the channel firmware to disable all network connections, the host application clears the Bus On flag in the "Application Change of State" register in the channel's control block. The firmware then closes all open network connections. A slave protocol stack would reject attempts to reopen a connection until the application allows opening network connections again (Bus On flag is set). When all connections are closed, the channel firmware clears the Communicating flag in the netX Communication Flags register.

Reset Command

System Reset vs. Channel Initialization

There are several methods to restart the netX firmware. The first is called "System Reset." The System Reset affects the netX operating system, rcX, and the protocol stacks. It forces the chip to immediately stop all running protocol stacks and the rcX itself. During the system reset, the netX is performing an internal memory check and other functions to insure the integrity of the netX chip itself.

The Channel Initialization, as the second method, affects a communication channel only. The channel firmware then reads and evaluates the configuration settings (or SYCON.net database, if available) again. The operating system is not affected. There are no particular tests performed during a channel initialization.

A third method to reset the netX chip is called Boot Start. No firmware is started when a System Reset is executed with the boot start flag set. The netX remains in boot loader mode.

A System Reset, Channel Initialization, and boot start may cause all network connection to be interrupted immediately, regardless of their current state.



During a HW-Reset and the time when the 2nd stage loader starts the Firmware, the content of the dual port memory can be 0xFFFF or 0x0BAD for a short period of time.

When used with Turbo PMAC2 CPU, it is necessary to reset the COMX module for proper functionality after initial power up, cycle power, or a \$\$\$ or \$\$\$*** command.

Resetting netX through Dual-Port Memory

To reset the entire netX firmware, the host application has to set the HSF_RESET bit in the bHostSysFlags register to perform a system-wide reset and respectively the APP_COS_INIT flag for a channel initialization in the ulApplicationCOS variable in the control block of the channel. The system reset and the channel initialization are handled differently by the firmware (see above).

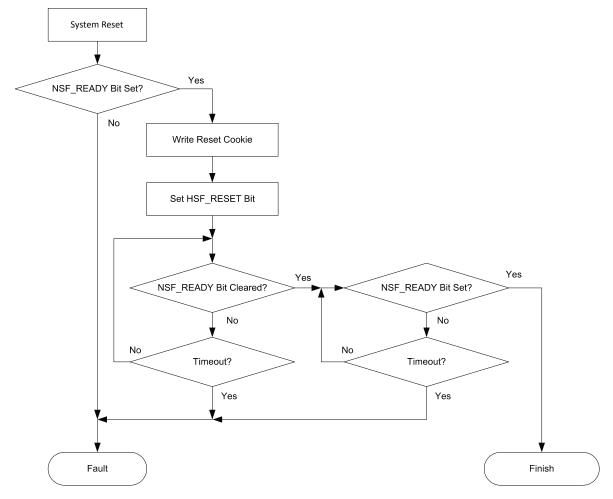
System Reset

To reset the netX operating system rcX and all communication channels, the host application has to write \$55AA55AA (System Reset Cookie) to the ulSystemCommandCOS variable in the system control block. Then, the HSF_RESET flag in bHostSysFlags has to be set. If the operating system does not find \$55AA55AA in the ulSystemCommandCOS variable, the reset command will be ignored.

The operating system clears the NSF_READY flag in bNetxFlags in the system handshake register, indicating that the system-wide reset is in progress. During the reset, all communication channel tasks are stopped, regardless of their current state. The rcX operating system flushes the entire dual-port memory and writes all memory locations to zero. After the reset, if rcX is finished without complications, and all protocol stacks are started properly, the NSF_READY flag is set again. Otherwise, the NSF_ERROR flag in bNetxFlags in the system handshake register is set, and an error code is written in ulSystemError in the system status block (see page 46), which helps identify possible problems.

Value	Definition/Description		
\$55AA55AA	System reset cookie		

The image below illustrates the steps the host application has to perform in order to execute a systemwide reset on the netX chip through the dual-port memory.



Timing

The duration of the reset outlined above depends on the firmware. Typically, the NSF_READY flag is cleared within around 100 - 500 ms after the HSF_RESET Flag was set. When cleared, the NSF_READY bit will be set again after around 0.5 - 5 s. Generally, the reset should not take more than 6 seconds.

Channel Initialization

In order to force the protocol stack to restart and evaluate the configuration parameter again, the application can set the APP_COS_INIT flag in the ulApplicationCOS register in the control block or send a reset packet to the communication channel. All open network connections are interrupted immediately, regardless of their current state. Reinitializing the channel is not allowed if the database is locked.

Changing flags in the ulApplicationCOS register requires the application also to toggle the host change of state command flag in the host communication flags register. Only then, the netX protocol stack recognizes the reset command.

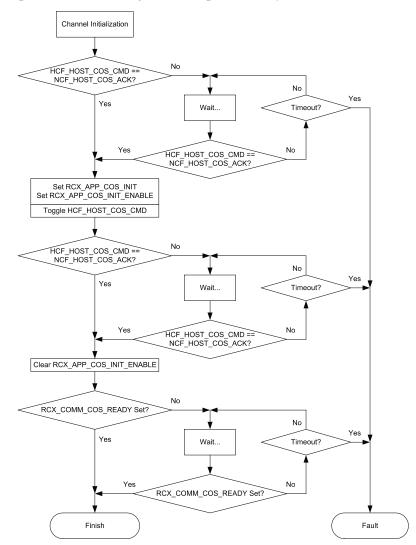
Below is the sequence:

CC0_RCX_APP_COS_INIT=1 CC0_RCX_APP_COS_INIT_ENABLE=1 HCCC0_HCF_HOST_COS_CMD=1

During channel initialization, the RCX_COMM_COS_READY flag and the RCX_COMM_COS_RUN flag are cleared together. The RCX_COMM_COS_READY flag stays cleared for at least 20 ms before it is set again, indicating that the initialization has finished. The RCX_COMM_COS_RUN flag is set if a valid configuration was found. Otherwise, it stays cleared.

After the initialization process has finished, the protocol stack checks ulApplicationCOS register. If the RCX_APP_COS_BUS_ON flag and the RCX_APP_COS_BUS_ON_ENABLE flags are set, network communication will be restored automatically. The same is true for the Lock Configuration feature (RCX_APP_COS_LOCK_CONFIG / RCX_APP_COS_LOCK_CONFIG_ENABLE) and the DMA data transfer mechanism (RCX_APP_COS_DMA / RCX_APP_COS_DMA_ENABLE).

The image below illustrates the steps the host application has to perform in order to execute a channel initialization on the protocol stack through the dual-port memory.



System Reset through Packets

The netX chip can be reset using a packet instead of the dual-port memory. The request packet is passed through the system mailbox. All open network connections are interrupted immediately, regardless of their current state. Reinitializing the channel is not allowed if the database is locked.

For detailed information about reset message settings, please see Hilscher documentation.

SOFTWARE SETUP

ACC-72EX supports multiple protocols, and setting up each protocol can be a bit different, as described in the protocol specific documentation provided by Hilscher. In this section, most of the generic steps are covered with the help of examples and screenshots.

Required Software Packages

Two software packages are required for setting up ACC-72EX:

- 1. SYCON.NET (V1.310.x.x or newer), available through Hilscher's website.
- 2. ACC-72EX Setup Assistant Software.

Both software packages have to be installed on the PC used for initial setup of the system and commissioning of the machine. Notice that neither of these software packages is required after the initial setup and the unit can work as a standalone setup.

SyCon.NET Software Setup

SYCON.net is a tool for the configuration of Fieldbus and Real-Time Ethernet systems. It is based on the standardized FDT / DTM technology. Online diagnostic indicators and auto-scan function for the reading of network participants assist in the commissioning of the network. SYCON.NET is provided with the gateway module under license from Hilscher Corporation.

With the power off, plug the ACC-72EX into the UBUS backplane and turn on the power to the UMAC rack. Connect the diagnostic port to a USB port on the PC using a micro-USB type cable. Launch the SYCON.NET software on the PC.



Administrator

OK

-

Cancel

User Name:

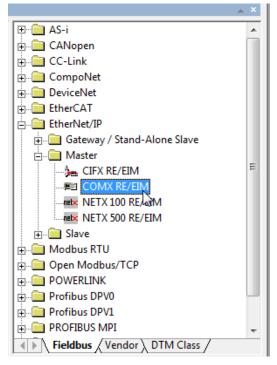
Password:

Enter the password:

SYCON.net - [Untitled.spj]		
<u>File View D</u> evice Ne <u>t</u> work E <u>x</u> tras <u>H</u> elp	<i>N</i>	
D 🗳 🖬 Q 4 4 5 😪 5: 🚳 🖪 5: 5; 5; 5;		
netProject A X		<u>⊾</u> ×
Project: Untitled	ASI CANopen CC-Link CompoNet CompoNet DeviceNet EtherCAT EtherNet/IP Open Modbus/TCP POWERLINK Profibus DPV0 Profibus DPV1 Profibus DPV1 PROFIBUS MPI PROFIBUS MPI FROFINET IO SERCOS III Fieldbus (Vendor) DTM Class / AS-i	
N TODO SYCON.net / netDevice /	 Administrator	

Start a new project or load an existing project from the File menu:

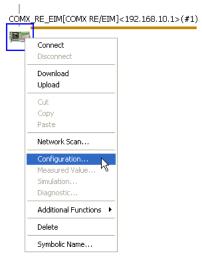
Select the COMX module to which the USB is connected from the Fieldbus protocol list. In this example, an EtherNet/IP module has been selected:



Drag and drop the module onto the BusLine in the netDevice window (notice that the module can only be inserted on the BusLine).

netDevice	
	^
netDevice	
COMX_RE_EIM[COMX_RE/EIM]<192.168.10.1>(#1)	
< >	

Establish USB communication to the COMX gateway by right-clicking on the device icon and selecting "Configuration...":

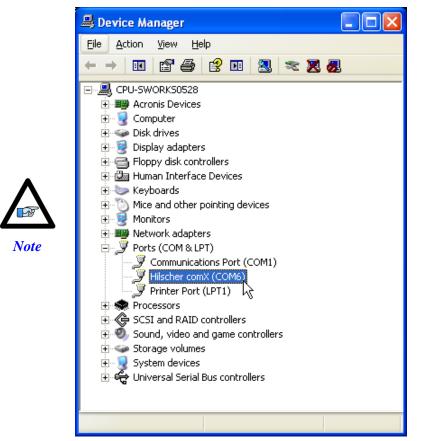


In the netDevice Configuration window, select the Driver folder under Settings folder in the NavigationArea, check the checkmark box for netX Driver on the driver list, and click Apply:

RetDevice - Configuration COMX_RE_EIM[COMX RE/EIM]<192.168.10.1>(#1)						
	KRE/EIM Ner GmbH		ice ID: 0x0104 dor ID: 0x011B	FDT		
Navigation Area 🚍						
Settings	Driver	Version	ID			
netX Driver	35Gateway Driver for n	etX (V3.x) 0.9.1.2	{787CD3A9-4CF6-4259-8E4D-1	09B6A6BEA91}		
Device Assignment	netX Driver	1.101.1.5347	{B54C8CC7-F333-4135-8405-66	E12FC88EE62}		
Firmware Download						
Configuration						
Network Settings						
Scanlist						
Process Data Address Table						
Scanner Settings						

Select the netXDriver node under the Driver folder in the Navigation Area and select the port resembling the USB connection to the COMX module. Click Save and Apply (just click OK if Apply is grayed out).

屠 netDevice - Configurati	on COMX_RE_EIM[COMX RE/EIM]<192.168	3.10.1>(#1)	
	IX RE/EIM her GmbH	Device ID: 0x0104 Vendor ID: 0x0118	FDT
Navigation Area 📃			
 Settings Driver netX Driver Device Assignment Firmware Download Configuration Network Settings Scanlist Process Data Address Table Scanner Settings 	USB/RS232 Connection TCP Connection F Enable USB/RS232 Connector (Restart of OD) Select Port: COM6 Port Configuratio COM1 Disable Port Baud Rate: 115.2 kBit/s Stop Bits: 1 Stopbit Send Timeout: 1000 mms Reset Timeout: 1000 mms		5
		Restore Save	Save All
		OK Cancel Apply	Help
\$)⊳ Disconnected 🚺 Data Set			



Check Windows Device Manager in order to identify which COM port provides the connection to the Hilscher COMX module.

Click Device Assignment under the Driver folder in the Navigation Area. Assign the netX Driver to the detected COMX module by checking the checkmark box next to the detected device, and click Apply.

kan ali netDevice - Configurati	ion COMX_RE_E	IM[COMX RE/EII	d]<192	.168.10	.1>(#1)		
	MX RE/EIM cher GmbH				Device ID: Vendor ID:	0x0104 0x011B	FDT
Navigation Area 📃							
Settings	Scan progress: 2/	2 Devices (Current	device: -)			
netX Driver Device Assignment Firmware Download	l D <u>e</u> vice selection:	suitable only	• •				Scan
Configuration	Device	Hardware Por	Slot	Serial	Driver	Channel Protocol	Access path
Network Settings Scanlist Process Data Address Table Scanner Settings	COMX	Ethernet/Ethe	n/a	21456	netX Driver	EtherNet/IP Scanner	\COM6
OK Cancel Apply Help							
🗘 Disconnected 🛛 🚺 Data Sel	t 🥖						

When used with Turbo PMAC, the reset line is released too fast for some Hilscher COMX modules, which puts them in a boot mode. This can prevent the device from being detected by Sycon.NET software. Make sure the device receives a system-wide reset using the PMAC suggested M-Variables ulSystemCommandCOS and HSF_RESET registers as shown here.

Note

SCtrl_ulSystemCommandCOS=\$55AA55AA

HCSC_HSF_RESET=1

Note that ACC-72EX Setup Assistant software automatically resets the cards if it cannot detect the identification cookie.

The rest of the steps are protocol/module dependent, and it is strongly recommended to follow the directions for these modules in Hilscher documentation available through their website. The current example will be continued with specifics to EtherNet/IP Scanner/Adapter setup.

Now that the COMX driver for communication between the PC and COMX module using the diagnostic port has been set up, go through protocol specific setup parameters under the Configuration folder in the Navigation Area.

RetDevice - Configurat	ion COMX_RE_EIM[COM)	(RE/EIM]<192.168.10).1>(#1)		
	MX RE/EIM scher GmbH		Device ID: Vendor ID:	0x0104 0x011B	FDT
Navigation Area 📃					
Settings Triver netX Driver	Description:	<_RE_EIM			
Device Assignment Firmware Download	IP Settings				
Configuration	DHCP				
Network Settings Scanlist	EootP				
Process Data Address Table	Fixed Addresses				
Scanner Settings	IP Address:	192 . 168 . 1	0.1		
	<u>N</u> etwork Mask:	255 . 255 . 25	55 . 0		
	Gateway Address:	0.0.0), 0		
	Note: The priority sequer	nce is DHCP, BootP, Fixed.			
	Operation mode: All car	pable, Auto Negotiation enal	bled		•
	operation interest [Fill cot	abio, Hato Nogotiation ona	biod		
			ОК	Cancel Apply	Help
∜⊳ Disconnected 🚺 Data Se	et 🗌				

After finishing modifying the settings for the device, press the OK button.

Back in the netDevice tree, right click on the device icon, and select Connect (as shown below).

netDevice			
			^
	:OMX RE/EIM]<192.168	10.15(#1)	
	OWN REFEINIS 192,100.	10.1>(#1)	
			-
COMX	_RE_EIS[COMX RE/EIS]<	:192.168.10.2>	
(III)			
	Connect		
-			
	Download Upload		
-	Cut		
	Сору		
_	Paste		
	Configuration		
	Measured Value		
	Simulation		
-	Diagnostic		
	Additional Functions 🔸		
	Delete		
	Symbolic Name		
<		>	~

Once connected, right click on the device icon one more time and select Download (as shown below). This will download all the configurations from PC to COMX module.

COMX_R	E_EIM	I[COMX RE/EIM]<192.168.10.1>(#1)
	CON	IX_RE_EIS[COMX RE/EIS]<192.168.10.2>
	time and	
		Connect
		Disconnect
		Download N
		Upload
		Cut
		Сору
		Paste
		Configuration
		Measured Value
		Simulation
		Diagnostic
		Additional Functions 🔸
		Delete
		Symbolic Name

Once the configuration is downloaded to the COMX module, make sure to save the SYCON.net project for later use.

The Hilscher slave module (COMX_RE_IES) above was dragged and dropped from the fieldbus protocol list. Third party slave modules can be added to that list by going to "Import Device Descriptions..." in the Network tab:

File View Device	Net	work	Extras	Help	
🗅 🚅 🔚 😰 🖣	랔	Add	Busline		
netProject		Delet	te Last Bu	Isline	
E 💼 Project: Untitled 🟃 Start Project Debug Mode					
COMX_RE_EI		Stop	Project [)ebug Mode	
		Devi	ce Catalo	g	
		Impo	ort Device	Descriptions	
		Print	Project l	Data	5

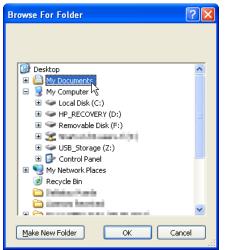
See Appendix C for an example setup using an ACC-72EX Ethernet IP slave with a third party Ethernet IP master PLC controller.

ACC-72EX Setup Assistant

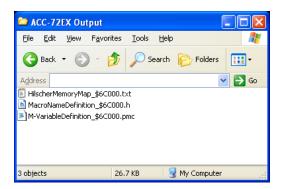
The next step is to generate the memory map and suggested M-Variables for the Hilscher module. Run the ACC-72EX Setup Assistant. In the Memory Map Generator groupbox, click the Connect to PMAC button, and select the UMAC where the ACC-72EX is installed. Once connected, the software will detect any available ACC-72EX(s) in the rack and list it based upon the base address(es).

ACC-72EX Setup Assistant	
Memory Map Generator	Address Converter
Connect to PMAC	ACC-72EX Base Address \$6C000
	Hilscher Address Offset 0x0
Address Selection \$6C000	Hilscher Data Width 32 💌
Starting M-Variable Number 6000 🗢	Hilscher Data Start Bit 0
	Convert
Generate M-Variable Definitions	Equivalent PMAC Address D:\$6C000
No PMAC is selected for communication. Connection to PMAC was successful. 2x ACC-72EX cards detected.	
Connected	

Select the starting number for M-Variable assignment, and click the "Generate M-Variable Definitions" button. The program asks for a folder location to save the M-Variable definition and memory map files.



This will generate three files which are named based upon the ACC-72EX base address.



The M-variable definition and its header file can be used in writing PLCs and motion programs in PMAC. The memory map file is useful for identifying the process data image locations.

Turbo PMAC Setup for Using ACC-72EX

All interactions between PMAC and ACC-72EX occur through M-variables. Most of the important registers which are required are mapped in the suggested M-Variable definition files generated by ACC-72EX Setup Assistant software. The generated files can be included in the header section of the project files in PEWIN32PRO2.



If multiple ACC-72EX cards are in the same UMAC rack, the Mvariable macro names will be identical for both files, despite the file name difference based upon the ACC-72EX base address. Make sure to add a prefix or suffix to the macro names both in the header file and definition file in order to distinguish the proper macro names for different ACC-72EXs. Note that no macro name should be longer than 32 characters.

There are multiple steps in getting the COMX module working with the network/fieldbus. Some of these steps are protocol-specific, and it is recommended to follow the requirements based upon each protocol manual provided by Hilscher.

Initialization PLC

Recall that ACC-72EX requires a reset after each power up, power cycle, \$\$\$ (reset), or \$\$\$*** (factory default reset). This can be achieved with a startup (or initialization) PLC. Example:

```
CLOSE
END GAT
DEL GAT
#include "M-VariableDefinition_$6C000.pmc"
#include "M-VariableDefinition $74000.pmc"
#define CommErrorFlag P1
OPEN PLC 1 CLEAR
DISABLE PLC 2..31
                                              // Disable all other tasks
SCtrl ulSystemCommandCOS=$55AA55AA
                                              // Reset token for MASTER Unit
HCSC HSF RESET=1
                                              // Reset bit, token required for reset to complete
S_SCtrl_ulSystemCommandCOS=$55AA55AA
                                             // Reset token for SLAVE Unit
                                              // Reset bit, token required for reset to complete
S HCSC HSF RESET=1
CommErrorFlag=0
timer = 1000 msec
                                              // Reset Time-out Timer
WHILE (CommErrorFlag=0 AND HCSC_NSF_READY=0) // Wait for reset to complete
       IF (timer<0)
                                              // Check for reset timeout
               CommErrorFlag = 1
       ENDIF
ENDWHILE
                                                             11
IF (CommErrorFlag=0)
       WHILE (CC0 RCX COMM COS RUN=0 OR S CC0 RCX COMM COS RUN=0)
                                                                     // wait for comm tasks to
                                                                     //\ {\rm start} on COMX modules
                       HCCC0 HCF NETX COS ACK = HCCC0 HCF NETX COS ACK ^ 1
                       // Toggle Communication Channel 0's Change of State Acknowledge bit in
                       // order to read the CCO RCX COMM COS RUN which is a part of Communication
                       // Channel 0 State Register
                      S HCCCO HCF NETX COS ACK = S HCCCO HCF NETX COS ACK ^ 1
       ENDWHILE
       ENABLE PLC 28
       ENABLE PLC 10
       ENABLE PLC 11
ENDIF
DISABLE PLC 1
CLOSE
```

Watchdog Function

The host Watchdog and the device Watchdog cells in the control block of each of the communication channels allow the operating system running on the netX to supervise the host or UMAC application and vice versa. There is no Watchdog function for the system block or for the handshake channel. The Watchdog for the channels is located in the control block of the status block of each communication channel.

The netX firmware reads the contents of the device Watchdog cell, increments the value by one, and copies it back into the host Watchdog location. Then, the application has to copy the new value from the host Watchdog location into the device Watchdog location. Copying the host Watchdog cell to the device Watchdog cell has to happen in the configured Watchdog time. When the overflow occurs, the firmware starts over and "1" appears in the host Watchdog cell. A zero turns off the Watchdog and therefore never appears in the host Watchdog cell in the regular process.

The minimum Watchdog time is 20 ms. The application can start the Watchdog function by copying any value unequal to zero into device Watchdog cell. A zero in the device Watchdog location stops the Watchdog function. The Watchdog timeout is configurable in SYCON.net and can be downloaded to the netX firmware.

If the application fails to copy the value from the host Watchdog location to the device Watchdog location within the configured Watchdog time, the protocol stack will interrupt all network connections immediately, regardless of their current state. If the Watchdog tripped, then power cycling, channel reset, or channel initialization will allow the communication channel to open network connections again.

Here is sample code for copying the host Watchdog location to the device Watchdog location:

```
CLOSE

END GAT

DEL GAT

#include "M-VariableDefinition_$6C000.pmc"

#include "M-VariableDefinition_$74000.pmc"

OPEN PLC 28 CLEAR

CC0_ulDeviceWatchdog = CC0_ulHostWatchdog // copies the host Watchdog content

// to device Watchdog cell

// for the 1st ACC-72EX

S_CC0_ulDeviceWatchdog = S_CC0_ulHostWatchdog // copies the host Watchdog content

// to device Watchdog cell

// for the 2nd ACC-72EX

CLOSE
```

Enabling the Communication Bus

Using the Bus On flag (CCx_RCX_APP_COS_BUS_ON, where x is the communication channel number), the host or UMAC application allows or disallows the netX firmware to open network connections. This flag is used together with the Bus On Enable flag

(CCx_RCX_APP_COS_BUS_ON_ENABLE, where x is the communication channel number). If set, the netX firmware tries to open network connections; if cleared, no connections are allowed, and open connections are closed. If the Bus On Enable flag is set, it enables the execution of the Bus On command in the netX firmware:

CC0_RCX_APP_COS_BUS_ON=1 CC0_RCX_APP_COS_BUS_ON_ENABLE=1	<pre>// Setting the Bus On flag for 1st ACC-72EX // Enabling the execution of Bus On Flag for 1st ACC-72EX</pre>
<pre>S_CC0_RCX_APP_COS_BUS_ON=1 S_CC0_RCX_APP_COS_BUS_ON_ENABLE=1</pre>	// Setting the Bus On flag for 2nd ACC-72EX // Enabling the execution of Bus On Flag for 2nd ACC-72EX

Locating the Input/Output Data Image in PMAC

Although the ACC-72EX Setup Assistant software defines M-Variables for accessing setup registers and flags in COMX modules, it does not assign any M-Variables for input/output data images. However, starting address and size of each input/output processed data image in's PMAC memory addressing format are calculated and included as a part of the memory map file that is generated. The following is an example from an EtherNet/IP option. The highlighted sections show the addressing for the processed data images:

+	Block 2:	
	Channel Type:	Communication
	Size of Channel:	15616 bytes
	Channel Start Address:	\$6C0C0
	Position of Handshake (Cells: IN HANDSHAKE CHANNEL
	Size of Handshake Cells	
	NetX Handshake Register	Y:\$6C082,0,16
	Host Handshake Register	x:\$6C082,0,16
	Communication Class:	SCANNER
	Protocol Class:	IO-DEVICE
	Conformance Class:	0
	Number of Subblocks:	9
	Subblock 0: CONTROL	
	Size:	8 bytes
	Start Offset:	\$6C0C2
	Transfer Direction:	OUT (Host System to netX)
	Transfer Type:	DPM (Dual-Port Memory)
	Handshake Mode:	UNCONTROLLED
	Handshake Bit:	0
	l	
	Subblock 1: COMMON STA	
	Size:	64 bytes
	Start Offset:	\$6C0C4
		IN (netX to Host System)
		DPM (Dual-Port Memory)
	Handshake Mode:	
	Handshake Bit:	0
	 Subblock 2: EXTENDED &	STATUS
	Size:	432 bytes
	Start Offset:	\$6C0D4
		IN (netX to Host System)
		DPM (Dual-Port Memory)
	Handshake Mode:	UNCONTROLLED
	Handshake Bit:	0
	Subblock 3: MAILBOX	1600 bytes
	Size: Start Offset:	1600 bytes \$6C140
		OUT (Host System to netX)
	Transfer Type:	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED
	Handshake Mode:	4
		-
	Subblock 4: MAILBOX	
	Size:	1600 bytes
	Start Offset:	\$6C2D0
	Transfer Direction:	IN (netX to Host System)
	Transfer Type:	DPM (Dual-Port Memory)
	Handshake Mode:	UNKNOWN
	Handshake Bit:	5
	Subblock 5: <mark>PROCESS D</mark>	
	Size:	5760 bytes
	Start Offset:	\$6C4C0
		OUT (Host System to netX)
	Transfer Type:	DPM (Dual-Port Memory)
		BUFFERED, HOST CONTROLLED
	Handshake Bit:	6

```
--- Subblock 6: PROCESS DATA IMAGE
        Size: 5760 bytes
Start Offset: $662460
        Size:
Transfer Direction: IN (netX to Host System)
        Transfer Type:DPM (Dual-Port Memory)Handshake Mode:BUFFERED, HOST CONTROLHandshake Bit:7
                                 BUFFERED, HOST CONTROLLED
|--- Subblock 7: HIGH PRIORITY DATA IMAGE
        Size: 64 bytes
Start Offset: $6C460
        Transfer Direction: OUT (Host System to netX)
        Transfer Type: DPM (Dual-Port Memory)
Handshake Mode: BUFFERED, HOST CONTROLLED
Handshake Bit: 8
        Handshake Bit:
|--- Subblock 8: HIGH PRIORITY DATA IMAGE
        Size: 64 bytes
Start Offset: $6C470
        Size:
        Transfer Direction: IN (netX to Host System)
        Transfer Type: DPM (Dual-Port Memory)
Handshake Mode: BUFFERED, HOST CONTROLLED
        Handshake Bit:
                                 9
```

Depending on the protocol, users might be interested in:

- Processed Data Images
- High Priority Data Images
- Mailboxes

Also listed in the memory map are starting address, size of each of these memory blocks, handshake method, and flag.

Reading/Writing from/to Input/Output Data Images

There are two methods for accessing processed data images:

1. Direct M-Variable definition to each register

This method is useful if the number of I/O data variables is small enough

2. Indirect M-Variable access

This method is mostly used if the number of I/O data count is greater than a comfortable level which can be handled by the direct M-Variable definition method. Refer to the Turbo PMAC Users Manual for detailed information on how to utilize the indirect addressing method.

This example demonstrates a 16-bit integer register transfer. Notice that only the first 16-bit portion of the integer in P200 will be transferred.

CLOSE

```
END GAT
DEL GAT
#include "M-VariableDefinition $6C000.pmc"
#include "M-VariableDefinition_$74000.pmc"
#define Master_OutputData1
                               M2000
#define Master InputData1
                             M2001
#define Slave_OutputData1
#define Slave_InputData1
                               M2002
                              M2003
Master OutputData1->Y:$6C4C0,0,16,S
                                              // Pointer to byte 0 and 1 of Output Data Image
                                               // of Communication Channel 0 on Master COMX module
Master InputData1->Y:$6CA60,0,16,S
                                              // Pointer to byte 0 and 1 of Input Data Image
                                              // of Communication Channel 0 on Master COMX module
Slave OutputData1->Y:$744C0,0,16,S
                                               // Pointer to byte 0 and 1 of Output Data Image
                                              // of Communication Channel 0 on Slave COMX module
                                              // Pointer to byte 0 and 1 of Input Data Image
Slave_InputData1->Y:$74A60,0,16,S
                                              // of Communication Channel 0 on Slave COMX module
P200=0
OPEN PLC 10 CLEAR
IF (HCCC0 HCF PD0 OUT CMD = HCCC0 NCF PD0 OUT ACK)
                                                      // Making sure the ACK flag matches the CMD
                                                       // flag before writing the value to the
                                                       // output data image register
        P200=P200+1
       Master_OutputData1 = P200
                                                       // Copy the value to register
                                                              // Toggle the CMD flag (^: XOR)
       HCCC0 \overline{\text{HCF}} PD0 OUT CMD = HCCC0 HCF PD0 OUT CMD^1
ENDIF
CLOSE
```

In a similar approach the data can be read from an input data image:

```
OPEN PLC 11 CLEAR

IF (HCCC0_NCF_PD0_IN_CMD = HCCC0_HCF_PD0_IN_ACK) // If CMD flag and ACK flags are

// equal, then the input data image

// register can be read

// read the input data image register

// toggle the acknowledge bit

// toggle the acknowledge bit

// indicating read completion
```

Notice that depending on M-Variable definition, different types of data formats can be transferred over the DPR and network:

Power PMAC Setup for Using ACC-72EX

Power PMAC has full support for ACC-72EX and all its fieldbus communication variations. Due to builtin data structures for accessing ACC-72EX dual ported RAM from Power PMAC, no additional software is required for memory mapping and/or identification in comparison to Turbo PMAC.

This section of the manual covers Power PMAC's built in data structures for ACC-72EX in addition to providing examples for header files, start-up and handshaking PLCs.

ACC72EX[i]. Non-Saved Data Structures

All of the interactions with ACC-72EX can be achieved through data structures defined specifically for ACC-72EX in Power PMAC firmware. The following structures allow access to the DPRAM in bit, byte, 2-byte and 4-byte wide access modes. The bit-wise read and write is only supported through Acc72EX[*i*].Udata16[*j*] data structure.

Acc72EX[*i*].Data8[*j*]

Description: Dual Ported RAM "unsigned 8-bit integer" data array element

Range: $0 .. 2^8 - 1$

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Data8[*j*] is the "*j*th" unsigned 8-bit integer data array element in the Acc72EX[*i*] dual-ported RAM. Each of these elements occupies one byte in the DPRAM, and is located starting at *j* addresses past the beginning of the buffer (which is located at the address in Acc72EX[*i*].a). This array is defined based upon the Hilscher ComX memory map.

Index values j in the square brackets can be integer constants in the range 0 to 524,287, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Data8[*j*] is located in the same registers as Acc72Ex[*i*].Idata16[*j*/2], Acc72Ex[*i*].Udata16[*j*/2], Acc72Ex[*i*].Idata32[*j*/4], Acc72Ex[*i*].Idata32[*j*/4] and Acc72Ex[*i*].Udata32[*j*/4]. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetData8 and ACC72EX SetData8 described later in this manual.

Acc72EX[i].Idata16[j]

Description: Dual Ported RAM "signed 16-bit integer" data array element

Range: $-2^{15} .. 2^{15} -1$

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Idata16[*j*] is the "*j*th" signed 16-bit integer data array element in the Acc72EX[*i*] dual-ported RAM. Each of these elements occupies two bytes in the DPRAM, and is located starting at 2**j* addresses past the beginning of the buffer (which is located at the address in Acc72EX[*i*].a). This array is defined based upon the Hilscher ComX memory map.

Index values j in the square brackets can be integer constants in the range 0 to 262,143, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Idata16[*j*] is located in the same registers as Acc72Ex[*i*].Data8[2**j*] to Acc72Ex[*i*].Data8[2**j*+1], Acc72Ex[*i*].Udata16[*j*], Acc72Ex[*i*].Idata32[*j*/2] and Acc72Ex[*i*].Udata32[*j*/2]. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetIdata16 and ACC72EX_SetIdata16 described later in this manual.

Acc72EX[*i*].Udata16[*j*]

Description: Dual Ported RAM "unsigned 16-bit integer" data array element

Range: $0 .. 2^{16} - 1$

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Udata16[*j*] is the "*j*th" unsigned 16-bit integer data array element in the Acc72EX[*i*] dualported RAM. Each of these elements occupies two bytes in the DPRAM, and is located starting at 2**j* addresses past the beginning of the buffer (which is located at the address in Acc72EX[*i*].a). This array is defined based upon the Hilscher ComX memory map.

Index values j in the square brackets can be integer constants in the range 0 to 262,143, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Udata16[*j*] is located in the same registers as Acc72Ex[*i*].Data8[2**j*] to Acc72Ex[*i*].Data8[2**j*+1], Acc72Ex[*i*].Idata16[*j*], Acc72Ex[*i*].Idata32[*j*/2] and Acc72Ex[*i*].Udata32[*j*/2]. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetUdata16 and ACC72EX_SetUdata16 described later in this manual.

Acc72EX[i].Idata32[j]

Description: Dual Ported RAM "signed 32-bit integer" data array element

Range: $-2^{31} .. 2^{31} - 1$

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Idata32[*j*] is the "*j*th" signed 32-bit integer data array element in the Acc72EX[*i*] dual-ported RAM. Each of these elements occupies four bytes in the DPRAM, and is located starting at 4**j* addresses past the beginning of the buffer (which is located at the address in Acc72EX[*i*].a). This array is defined based upon the Hilscher ComX memory map.

Index values j in the square brackets can be integer constants in the range 0 to 131,072, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Idata32[*j*] is located in the same registers as Acc72Ex[*i*].Data8[4**j*] to Acc72Ex[*i*].Data8[4**j*+5], Acc72Ex[*i*].Idata16[2**j*] to Acc72Ex[*i*].Idata16[2**j*+1], Acc72Ex[*i*].Udata16[2**j*] to Acc72Ex[*i*].Udata16[2**j*+1] and Acc72Ex[*i*].Udata32[*j*]. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetIdata32 and ACC72EX_SetIdata32 described later in this manual.

Acc72EX[*i*].Udata32[*j*]

Description: Dual Ported RAM "unsigned 16-bit integer" data array element

Range: $0 .. 2^{32} - 1$

Units: address dependent

Power-on default: address dependent

Acc72Ex[*i*].Udata32[*j*] is the "*j*th" unsigned 32-bit integer data array element in the Acc72EX[*i*] dual ported RAM. Each of these elements occupies four bytes in the DPRAM, and is located starting at 4**j* addresses past the beginning of the buffer (which is located at the address in Acc72EX[*i*].a). This array is defined based upon the Hilscher ComX memory map.

Index values j in the square brackets can be integer constants in the range 0 to 262,143, or local L-variables. No expressions or non-integer constants are permitted. The size of the DPRAM is dependent on the ACC-72EX communication option and installed Hilscher ComX module.

Acc72Ex[*i*].Udata32[*j*] is located in the same registers as Acc72Ex[*i*].Data8[4**j*] to Acc72Ex[*i*].Data8[4**j*+5], Acc72Ex[*i*].Idata16[2**j*] to Acc72Ex[*i*].Idata16[2**j*+1], Acc72Ex[*i*].Udata16[2**j*] to Acc72Ex[*i*].Udata16[2**j*+1] and Acc72Ex[*i*].Idata32[*j*]. It is the user's responsibility to prevent possible multiple uses of the same register.

In C, this element should be accessed through the C functions ACC72EX_GetUdata32 and ACC72EX_SetUdata32 described later in this manual.

C Programming Access to ACC-72EX Structures

One can use the following header file full of functions to read from and write to the aforementioned Acc72EX[i] structures from a C program. The input argument CardIndex is *i* and ArrayIndex is *j* as above. Use the "Get" functions to retrieve the structure values; use the "Set" functions to write to the structures. In the "Set" functions, the Input argument is the value to which to set the structure.

```
int Acc72EX GetIdata32(unsigned int CardIndex, unsigned int ArrayIndex);
unsigned int Acc72EX GetUdata32(unsigned int CardIndex, unsigned int ArrayIndex);
short Acc72EX GetIdata16(unsigned int CardIndex, unsigned int ArrayIndex);
char Acc72EX_GetData8(unsigned int CardIndex, unsigned int ArrayIndex);
unsigned short Acc72EX GetUdata16(unsigned int CardIndex, unsigned int ArrayIndex);
void Acc72EX SetIdata16(unsigned int CardIndex, unsigned int ArrayIndex, short Input);
void Acc72EX_SetUdatal6(unsigned int CardIndex, unsigned int ArrayIndex, unsigned short Input);
void Acc72EX_SetIdata32(unsigned int CardIndex, unsigned int ArrayIndex, int Input);
void Acc72EX SetUdata32 (unsigned int CardIndex, unsigned int ArrayIndex, unsigned int Input);
void Acc72EX SetData8(unsigned int CardIndex, unsigned int ArrayIndex, char Input);
short Acc72EX GetIdata16(unsigned int CardIndex, unsigned int ArrayIndex)
       unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
       return (short)((myptr[ArrayIndex] << 8) >> 16);
unsigned short Acc72EX GetUdata16(unsigned int CardIndex, unsigned int ArrayIndex)
{
       unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
       return (unsigned short)((myptr[ArrayIndex] << 8) >> 16);
}
char Acc72EX GetData8 (unsigned int CardIndex, unsigned int ArrayIndex)
{
       unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
return (myptr[ArrayIndex / 2] << (16 / (1 + (ArrayIndex % 4) % 2))) >> 24;
unsigned int Acc72EX GetUdata32(unsigned int CardIndex, unsigned int ArrayIndex)
{
       unsigned int i = ArrayIndex * 4, j, k = 0;
       unsigned int out = 0;
        for(j = i; j <= i + 3; j++)</pre>
       {
               out |= (unsigned int)((unsigned int)Acc72EX GetData8(CardIndex, j) << (8 * k));</pre>
               k++;
       }
       return out:
int Acc72EX GetIdata32 (unsigned int CardIndex, unsigned int ArrayIndex)
{
       return (int)Acc72EX_GetUdata32(CardIndex, ArrayIndex);
}
void Acc72EX SetIdata16(unsigned int CardIndex, unsigned int ArrayIndex, short Input)
{
       unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
       myptr[ArrayIndex] = (Input << 8) & 0x00FFFF00;</pre>
void Acc72EX SetUdata16(unsigned int CardIndex, unsigned int ArrayIndex, unsigned short Input)
       unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
       myptr[ArrayIndex] = (Input << 8) & 0x00FFFF00;</pre>
```

```
void Acc72EX_SetIdata32(unsigned int CardIndex, unsigned int ArrayIndex, int Input)
{
    unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
    myptr[ArrayIndex] = ((Input << 16) >> 8);
    myptr[ArrayIndex + 1] = ((Input >> 16) << 8);
}
void Acc72EX_SetUdata32(unsigned int CardIndex, unsigned int ArrayIndex, unsigned int Input) {
    unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
    myptr[ArrayIndex] = (Input << 16) >> 8;
    myptr[ArrayIndex] = (Input << 16) >> 8;
    myptr[ArrayIndex + 1] = ((Input >> 16) << 8);
}
void Acc72EX_SetData8(unsigned int CardIndex, unsigned int ArrayIndex, char Input) {
    unsigned int *myptr = (unsigned int *)piom + (DPRCSBase + CardIndex * 0x100000) / 4;
    unsigned int shift = (8 * (1 + ArrayIndex % 2));
    unsigned int ind = ArrayIndex / 2;
    myptr[ind] &= ~(0x000000FF << shift);
    myptr[ind] |= (Input << shift);
}
</pre>
```

Global Header for Power PMAC Projects

This section provides example for header files which allow use of native netX variable names rather than using Power PMAC structures. The following header file is written as generically as possible allowing access to most used registers in System, Handshake and Communication Channels.

```
/* ACC-72EX Power PMAC Project Header
/\star This header file provides macro definitions for most common registers in Hilsche COMX modules
/* used in ACC-72EX.
/*
/* Instructions:
/* Uncomment the related \# define \ depending \ on \ ACC-72EX \ option
/*
/**
    //#define __PROFIBUS_DP_Master__
//#define __PROFIBUS_DP_Slave__
//#define __DeviceNet_Master__
//#define __DeviceNet_Slave__
//#define __CANopen_Master__
//#define __CANopen_Slave__
//#define __CC_Link_Slave_
//#define __EtherCAT_Master____
//#define __EtherCAT_Slave____
//#define __EtherNetIP_Scanner_Master___
//#define __EtherNetIP_Adapter_Slave__
// System Information Block Structure
        #define SI_abCookie_0_
                                                              Acc72Ex[0].Data8[0]
        #define SI_abCookie_1_
                                                              Acc72Ex[0].Data8[1]
        #define SI_abCookie_2_
                                                              Acc72Ex[0].Data8[2]
        #define SI abCookie 3
                                                              Acc72Ex[0].Data8[3]
        #define SI ulDpmTotalSize
                                                              Acc72Ex[0].Udata32[1]
        #define SI_ulDeviceNumber
                                                              Acc72Ex[0].Udata32[2]
        #define SI ulSerialNumber
                                                              Acc72Ex[0].Udata32[3]
        #define SI ausHwOptions 0
                                                              Acc72Ex[0].Udata16[8]
        #define SI_ausHwOptions_1_
                                                              Acc72Ex[0].Udata16[9]
        #define SI_ausHwOptions_2_
#define SI_ausHwOptions_3_
                                                              Acc72Ex[0].Udata16[10]
                                                              Acc72Ex[0].Udata16[11]
        #define SI usManufacturer
                                                             Acc72Ex[0].Udata16[12]
        #define SI_usProductionDate
#define SI_ulLicenseFlags1
                                                              Acc72Ex[0].Udata16[13]
                                                              Acc72Ex[0].Udata32[7]
        #define SI ulLicenseFlags2
                                                             Acc72Ex[0].Udata32[8]
        #define SI_usNetxLicenseID
                                                              Acc72Ex[0].Udata16[18]
        #define SI_usNetxLicenseFlags
#define SI_usDeviceClass
                                                              Acc72Ex[0].Udata16[19]
                                                              Acc72Ex[0].Udata16[20]
        #define SI bHwRevision
                                                              Acc72Ex[0].Data8[42]
        #define SI_bHwCompatibility
#define SI_bDevIdNumber
                                                              Acc72Ex[0].Data8[43]
                                                              Acc72Ex[0].Data8[44]
// System Channel Information Structure
        #define SCI bChannelType
                                                              Acc72Ex[0].Data8[48]
        #define SCI bSizePositionOfHandshake
                                                              Acc72Ex[0].Data8[50]
        #define SCI bNumberOfBlocks
                                                             Acc72Ex[0].Data8[51]
        #define SCI_ulSizeOfChannel
                                                              Acc72Ex[0].Udata32[13]
        #define SCI usSizeOfMailbox
                                                              Acc72Ex[0].Udata16[28]
        #define SCI_usMailboxStartOffset
                                                              Acc72Ex[0].Udata16[29]
// Handshake Channel Information Structure
        #define HCI_bChannelType
                                                              Acc72Ex[0].Data8[64]
        #define HCI ulSizeOfChannel
                                                              Acc72Ex[0].Udata32[17]
// Communication Channel 0 Information Structure
        #define CC0I bChannelType
                                                              Acc72Ex[0].Data8[80]
        #define CCOI bChannelId
                                                              Acc72Ex[0].Data8[81]
        #define CC01 bSizePositionOfHandshake
                                                              Acc72Ex[0].Data8[82]
```

<pre>#define CC0I_bNumberOfBlocks Acc72Ex[0].Data8[83] #define CC0I_ulSizeOfChannel Acc72Ex[0].Udata32[21] #define CC0I_usCommunicationClass Acc72Ex[0].Udata16[44] #define CC0I_usProtocolClass Acc72Ex[0].Udata16[45] #define CC0I_usConformanceClass Acc72Ex[0].Udata16[46] // Communication Channel 1 Information Structure #define CC1I_bChannelType Acc72Ex[0].Data8[96] #define CC1I_bChannelId Acc72Ex[0].Data8[97] #define CC1I_bSizePositionOfHandshake Acc72Ex[0].Data8[98]</pre>	
<pre>#define CC0I_ulSizeOfChannel Acc72Ex[0].Udata32[21] #define CC0I_usCommunicationClass Acc72Ex[0].Udata16[44] #define CC0I_usProtocolClass Acc72Ex[0].Udata16[45] #define CC0I_usConformanceClass Acc72Ex[0].Udata16[46] // Communication Channel 1 Information Structure #define CC1I_bChannelType Acc72Ex[0].Data8[96] #define CC1I_bChannelId Acc72Ex[0].Data8[97] #define CC1I_bSizePositionOfHandshake Acc72Ex[0].Data8[98]</pre>	
<pre>#define CC0I_usCommunicationClass Acc72Ex[0].Udata16[44] #define CC0I_usProtocolClass Acc72Ex[0].Udata16[45] #define CC0I_usConformanceClass Acc72Ex[0].Udata16[46] // Communication Channel 1 Information Structure #define CC1I_bChannelType Acc72Ex[0].Data8[96] #define CC1I_bChannelId Acc72Ex[0].Data8[97] #define CC1I_bSizePositionOfHandshake Acc72Ex[0].Data8[98]</pre>	
<pre>#define CC0I_usProtocolClass Acc72Ex[0].Udata16[45] #define CC0I_usConformanceClass Acc72Ex[0].Udata16[46] // Communication Channel 1 Information Structure #define CC1I_bChannelType Acc72Ex[0].Data8[96] #define CC1I_bChannelId Acc72Ex[0].Data8[97] #define CC1I_bSizePositionOfHandshake Acc72Ex[0].Data8[98]</pre>	
<pre>#define CC0I_usProtocolClass Acc72Ex[0].Udata16[45] #define CC0I_usConformanceClass Acc72Ex[0].Udata16[46] // Communication Channel 1 Information Structure #define CC1I_bChannelType Acc72Ex[0].Data8[96] #define CC1I_bChannelId Acc72Ex[0].Data8[97] #define CC1I_bSizePositionOfHandshake Acc72Ex[0].Data8[98]</pre>	
#define CC0I_usConformanceClass Acc72Ex[0].Udata16[46] // Communication Channel 1 Information Structure #define CC11_bChannelType #define CC11_bChannelId Acc72Ex[0].Data8[96] #define CC11_bSizePositionOfHandshake Acc72Ex[0].Data8[98]	
// Communication Channel 1 Information Structure #define CC11_bChannelType Acc72Ex[0].Data8[96] #define CC11_bChannelId Acc72Ex[0].Data8[97] #define CC11_bSizePositionOfHandshake Acc72Ex[0].Data8[98]	
#define CC11_bChannelTypeAcc72Ex[0].Data8[96]#define CC11_bChannelIdAcc72Ex[0].Data8[97]#define CC11_bSizePositionOfHandshakeAcc72Ex[0].Data8[98]	
#define CC11_bChannelTypeAcc72Ex[0].Data8[96]#define CC11_bChannelIdAcc72Ex[0].Data8[97]#define CC11_bSizePositionOfHandshakeAcc72Ex[0].Data8[98]	
#define CC11_bChannelTypeAcc72Ex[0].Data8[96]#define CC11_bChannelIdAcc72Ex[0].Data8[97]#define CC11_bSizePositionOfHandshakeAcc72Ex[0].Data8[98]	
<pre>#define CC11_bChannelId Acc72Ex[0].Data8[97] #define CC11_bSizePositionOfHandshake Acc72Ex[0].Data8[98]</pre>	
<pre>#define CC11_bSizePositionOfHandshake Acc72Ex[0].Data8[98]</pre>	
<pre>#define CC11_bSizePositionOfHandshake Acc72Ex[0].Data8[98]</pre>	
#define CC1I bNumberOfBlocks Acc72Ex[0].Data8[99]	
#define CC1I usCommunicationClass Acc72Ex[0].Udata16[52]	
#define CC11_usProtocolClass Acc72Ex[0].Udata16[53]	
<pre>#define CC11_usConformanceClass Acc72Ex[0].Udata16[54]</pre>	
// Communication Channel 0 Information Structure	
#define CC2I bChannelType Acc72Ex[0].Data8[112]	
#define CC2I bChannelId Acc72Ex[0].Data8[113]	
<pre>#define CC2I_bSizePositionOfHandshake Acc72Ex[0].Data8[114]</pre>	
#define CC2I bNumberOfBlocks Acc72Ex[0].Data8[115]	
#define CC2I usCommunicationClass Acc72Ex[0].Udata16[60]	
#define CC2I usProtocolClass Acc72Ex[0].Udata16[61]	
<pre>#define CC2I usConformanceClass Acc72Ex[0].Udata16[62]</pre>	
// Communication Channel 1 Information Structure	
// Communication channel i information Structure	
<pre>#define CC3I bChannelType Acc72Ex[0].Data8[128]</pre>	
#define CC3I bChannelId Acc72Ex[0].Data8[129]	
#define CC3I bSizePositionOfHandshake Acc72Ex[0].Data8[130]	
#define CC3I bNumberOfBlocks Acc72Ex[0].Data8[131]	
<pre>#define CC3I_ulSizeOfChannel Acc72Ex[0].Udata32[33]</pre>	
#define CC3I usCommunicationClass Acc72Ex[0].Udata16[68]	
#define CC3I_usProtocolClass Acc72Ex[0].Udata16[69]	
<pre>#define CC3I usConformanceClass Acc72Ex[0].Udata16[70]</pre>	
// Application Channel O Information Churchurg	
// Application Channel 0 Information Structure	
#define ACOI bChannelType Acc72Ex[0].Data8[144]	
#define ACOI bChannelId Acc72Ex[0].Data8[145]	
#define ACOI bSizePositionOfHandshake Acc72Ex[0].Data8[146]	
#define AC01 bNumberOfBlocks Acc72Ex[0].Data8[147]	
<pre>#define AC01_ulSizeOfChannel Acc72Ex[0].Udata32[37]</pre>	
// Application Channel 1 Information Structure	
#define AC1I bChannelType Acc72Ex[0].Data8[160]	
#define AC11 bChannelId Acc72Ex[0].Data8[161]	
	ł
<pre>#define AC11_bSizePositionOfHandshake Acc72Ex[0].Data8[162]</pre>	
#define AC11 bNumberOfBlocks Acc72Ex[0].Data8[163]	
#define AC1I_ulSizeOfChannel Acc72Ex[0].Udata32[41]	ŀ
Rectine Acti_atolizeotonamiet ACC/2EX[0].0dato2[41]	ŀ
	ŀ
// System Control Block Structure	
#define SCtrl ulSystemCommandCOS Acc72Ex[0].Udata32[46]	
#define Souri_utSystemcommandcos ACC/ZEX[0].0datd32[46]	
// System Status Block Structure	
#define SStat_ulSystemCOS Acc72Ex[0].Udata32[48]	
#define SStat ulSystemStatus Acc72Ex[0].Udata32[49]	
#define SStat ulSystemError Acc72Ex[0].Udata32[50]	
<pre>#define SStat_ulBootError Acc72Ex[0].Udata32[51]</pre>	
#define SStat_ulTimeSinceStart Acc72Ex[0].Udata32[52]	
#define SStat_usCpuLoad Acc72Ex[0].Udata16[106]	
<pre>#define SStat ulHWFeatures Acc72Ex[0].Udata16[108]</pre>	
// NETX_SYSTEM_SEND_MAILBOX	
// NETX_SYSTEM_SEND_MAILBOX #define SSMB_usPackagesAccepted Acc72Ex[0].Udata16[128]	
#define SSMB_usPackagesAccepted Acc72Ex[0].Udata16[128]	
#define SSMB_usPackagesAcceptedAcc72Ex[0].Udata16[128]#define SSMB_ulDestAcc72Ex[0].Udata32[65]	
#defineSSMB_usPackagesAcceptedAcc72Ex[0].Udata16[128]#defineSSMB_ulDestAcc72Ex[0].Udata32[65]#defineSSMB_ulSrcAcc72Ex[0].Udata32[66]	
#define SSMB_usPackagesAcceptedAcc72Ex[0].Udata16[128]#define SSMB_ulDestAcc72Ex[0].Udata32[65]	
#defineSSMB_usPackagesAcceptedAcc72Ex[0].Udata16[128]#defineSSMB_ulDestAcc72Ex[0].Udata32[65]#defineSSMB_ulSrcAcc72Ex[0].Udata32[66]#defineSSMB_ulDestIdAcc72Ex[0].Udata32[67]	
#defineSSMB_usPackagesAcceptedAcc72Ex[0].Udata16[128]#defineSSMB_ulDestAcc72Ex[0].Udata32[65]#defineSSMB_ulSrcAcc72Ex[0].Udata32[66]#defineSSMB_ulDestIdAcc72Ex[0].Udata32[67]#defineSSMB_ulSrcIdAcc72Ex[0].Udata32[68]	
#defineSSMB_usPackagesAcceptedAcc72Ex[0].Udata16[128]#defineSSMB_ulDestAcc72Ex[0].Udata32[65]#defineSSMB_ulSrcAcc72Ex[0].Udata32[66]#defineSSMB_ulDestIdAcc72Ex[0].Udata32[67]#defineSSMB_ulSrcIdAcc72Ex[0].Udata32[68]#defineSSMB_ulLenAcc72Ex[0].Udata32[69]	
#defineSSMB_usPackagesAcceptedAcc72Ex[0].Udata16[128]#defineSSMB_ulDestAcc72Ex[0].Udata32[65]#defineSSMB_ulSrcAcc72Ex[0].Udata32[66]#defineSSMB_ulDestIdAcc72Ex[0].Udata32[67]#defineSSMB_ulSrcIdAcc72Ex[0].Udata32[68]	

#define SSMB ulState	Acc72Ex[0].Udata32[71]
#define SSMB ulCmd	
-	Acc72Ex[0].Udata32[72]
#define SSMB_ulExt	Acc72Ex[0].Udata32[73]
#define SSMB ulRout	Acc72Ex[0].Udata32[74]
r = r = 0.000 $P = r = 0.000$ $> t = 0.000$	
<pre>ptr SSMB_Data8(84)->*;</pre>	
ptr SSMB Data16(42)->*;	
ptr SSMB Data32(21)->*;	
// NETX_SYSTEM_RECEIVE_MAILBOX	
#define SRMB_usWaitingPackages	Acc72Ex[0].Udata16[192]
_	
#define SRMB ulDest	Acc72Ex[0].Udata32[97]
#define SRMB_ulSrc	Acc72Ex[0].Udata32[98]
#define SRMB ulDestId	Acc72Ex[0].Udata32[99]
#define SRMB ulSrcId	Acc72Ex[0].Udata32[100]
#define SRMB_ulLen	Acc72Ex[0].Udata32[101]
_	
#define SRMB_ulId	Acc72Ex[0].Udata32[102]
#define SRMB ulState	Acc72Ex[0].Udata32[103]
#define SRMB_ulCmd	Acc72Ex[0].Udata32[104]
#define SRMB ulExt	Acc72Ex[0].Udata32[105]
_	
#define SRMB_ulRout	Acc72Ex[0].Udata32[106]
ptr SRMB Data8(84)->*;	
ptr SRMB Data16(42) ->*;	
ptr SRMB_Data32(21)->*;	
// SC bNetxFlags	
#define HCSC NSF READY	Acc72Ex[0].Udata16[257].0
·	
#define HCSC_NSF_ERROR	Acc72Ex[0].Udata16[257].1
#define HCSC_NSF_HOST_COS_ACK	Acc72Ex[0].Udata16[257].2
#define HCSC NSF NETX COS CMD	Acc72Ex[0].Udata16[257].3
#define HCSC NSF SEND MBX ACK	Acc72Ex[0].Udata16[257].4
#define HCSC_NSF_RECV_MBX_CMD	Acc72Ex[0].Udata16[257].5
// SC_bHostFlags	
#define HCSC HSF RESET	Acc72Ex[0].Udata16[257].8
#define HCSC HSF BOOTSTART	Acc72Ex[0].Udata16[257].9
#define HCSC HSF HOST COS CMD	Acc72Ex[0].Udata16[257].10
#define HCSC_HSF_NETX_COS_ACK	Acc72Ex[0].Udata16[257].11
#define HCSC HSF SEND MBX CMD	Acc72Ex[0].Udata16[257].12
#define HCSC HSF RECV MBX ACK	Acc72Ex[0].Udata16[257].13
// CCO usNetxFlags	
2	
#define HCCC0_usNetxFlags	Acc72Ex[0].Udata16[260]
#define HCCC0_NCF_COMMUNICATING	Acc72Ex[0].Udata16[260].0
#define HCCC0 NCF ERROR	Acc72Ex[0].Udata16[260].1
#define HCCC0 NCF HOST COS ACK	Acc72Ex[0].Udata16[260].2
#define HCCC0_NCF_NETX_COS_CMD	Acc72Ex[0].Udata16[260].3
#define HCCC0_NCF_SEND_MBX_ACK	Acc72Ex[0].Udata16[260].4
#define HCCC0 NCF RECV MBX CMD	Acc72Ex[0].Udata16[260].5
#define HCCC0 NCF PD0 OUT ACK	Acc72Ex[0].Udata16[260].6
#define HCCCO_NCF_PDO_IN_CMD	Acc72Ex[0].Udata16[260].7
#define HCCC0_NCF_PD1_OUT_ACK	Acc72Ex[0].Udata16[260].8
#define HCCC0_NCF_PD1_IN_CMD	Acc72Ex[0].Udata16[260].9
// CCO usHostFlags	
#define HCCC0 usHostFlags	Acc72Ex[0].Udata16[261]
#define HCCC0_HCF_HOST_COS_CMD	Acc72Ex[0].Udata16[261].2
#define HCCC0_HCF_NETX_COS_ACK	Acc72Ex[0].Udata16[261].3
#define HCCC0 HCF SEND MBX CMD	Acc72Ex[0].Udata16[261].4
#define HCCC0 HCF RECV MBX ACK	Acc72Ex[0].Udata16[261].5
#define HCCC0_HCF_PD0_OUT_CMD	Acc72Ex[0].Udata16[261].6
#define HCCC0_HCF_PD0_IN_ACK	Acc72Ex[0].Udata16[261].7
#define HCCCO HCF PD1 OUT CMD	Acc72Ex[0].Udata16[261].8
#define HCCC0 HCF PD1 IN ACK	Acc72Ex[0].Udata16[261].9
	100/2DA[0].00000010[201].9
// CC1 usNetxFlags	
#define HCCC1_usNetxFlags	Acc72Ex[0].Udata16[262]
#define HCCC1 NCF COMMUNICATING	Acc72Ex[0].Udata16[262].0
#define HCCC1 NCF ERROR	Acc72Ex[0].Udata16[262].1
#define HCCC1 NCF HOST COS ACK	
	Acc72Ex[0].Udata16[262].2
#define HCCC1_NCF_NETX_COS_CMD	Acc72Ex[0].Udata16[262].3
#define HCCC1 NCF SEND MBX ACK	Acc72Ex[0].Udata16[262].4
#define HCCC1 NCF RECV MBX CMD	Acc72Ex[0].Udata16[262].5
	100/2DA[0].00000010[202].0

#define HCCC1 NCF PD0 OUT ACK #define HCCC1 NCF PD0 IN CMD #define HCCC1_NCF_PD1_OUT_ACK #define HCCC1_NCF_PD1_IN_CMD // CC1 usHostFlags #define HCCC1 usHostFlags #define HCCC1_HCF_HOST_COS_CMD
#define HCCC1_HCF_NETX_COS_ACK #define HCCC1 HCF SEND MBX CMD #define HCCC1_HCF_RECV_MBX_ACK
#define HCCC1_HCF_PD0_OUT_CMD #define HCCC1 HCF PD0 IN ACK #define HCCC1_HCF_PD1_OUT_CMD #define HCCC1 HCF PD1 IN ACK // CC2 usNetxFlags #define HCCC2 usNetxFlags #define HCCC2_NCF_COMMUNICATING
#define HCCC2_NCF_ERROR #define HCCC2 NCF HOST COS ACK #define HCCC2 NCF NETX COS CMD #define HCCC2_NCF_SEND_MBX_ACK
#define HCCC2_NCF_RECV_MBX_CMD #define HCCC2 NCF PD0 OUT ACK #define HCCC2_NCF_PD0_IN_CMD #define HCCC2_NCF_PD1_OUT_ACK
#define HCCC2_NCF_PD1_IN_CMD // CC2 usHostFlags #define HCCC2_usHostFlags
#define HCCC2 HCF HOST COS CMD #define HCCC2 HCF NETX COS ACK #define HCCC2_HCF_SEND_MBX_CMD
#define HCCC2_HCF_RECV_MBX_ACK #define HCCC2 HCF PD0 OUT CMD #define HCCC2 HCF PD0 IN ACK #define HCCC2_HCF_PD1_OUT_CMD
#define HCCC2_HCF_PD1_IN_ACK // CC3 usNetxFlags #define HCCC3_usNetxFlags #define HCCC3 NCF COMMUNICATING #define HCCC3 NCF ERROR #define HCCC3 NCF HOST COS ACK #define HCCC3_NCF_NETX_COS_CMD
#define HCCC3_NCF_SEND_MBX_ACK #define HCCC3 NCF RECV MBX CMD #define HCCC3_NCF_PD0_OUT_ACK
#define HCCC3_NCF_PD0_IN_CMD
#define HCCC3_NCF_PD1_OUT_ACK #define HCCC3 NCF PD1 IN CMD // CC3 usHostFlags #define HCCC3 usHostFlags #define HCCC3 HCF HOST COS CMD #define HCCC3_HCF_NETX_COS_ACK #define HCCC3_HCF_SEND_MBX_CMD
#define HCCC3_HCF_RECV_MBX_ACK #define HCCC3 HCF PD0 OUT CMD #define HCCC3_HCF_PD0_IN_ACK
#define HCCC3_HCF_PD1_OUT_CMD #define HCCC3 HCF PD1 IN ACK // CCO Control Block #define CC0 RCX APP COS APP READY #define CCO RCX APP COS BUS ON #define CC0_RCX_APP_COS_BUS_ON_ENABLE
#define CC0_RCX_APP_COS_INIT #define CC0 RCX APP COS INIT ENABLE #define CC0 RCX APP COS LOCK CFG #define CC0_RCX_APP_COS_LOCK_CFG_ENA
#define CC0_RCX_APP_COS_DMA #define CC0 RCX APP COS DMA ENABLE #define CC0 ulDeviceWatchdog

Acc72Ex[0].Udata16[262].6 Acc72Ex[0].Udata16[262].7 Acc72Ex[0].Udata16[262].8 Acc72Ex[0].Udata16[262].9 Acc72Ex[0].Udata16[263] Acc72Ex[0].Udata16[263].2 Acc72Ex[0].Udata16[263].3 Acc72Ex[0].Udata16[263].4 Acc72Ex[0].Udata16[263].5 Acc72Ex[0].Udata16[263].6 Acc72Ex[0].Udata16[263].7 Acc72Ex[0].Udata16[263].8 Acc72Ex[0].Udata16[263].9 Acc72Ex[0].Udata16[264] Acc72Ex[0].Udata16[264].0 Acc72Ex[0].Udata16[264].1 Acc72Ex[0].Udata16[264].2 Acc72Ex[0].Udata16[264].3 Acc72Ex[0].Udata16[264].4 Acc72Ex[0].Udata16[264].5 Acc72Ex[0].Udata16[264].6 Acc72Ex[0].Udata16[264].7 Acc72Ex[0].Udata16[264].8 Acc72Ex[0].Udata16[264].9 Acc72Ex[0].Udata16[265] Acc72Ex[0].Udata16[265].2 Acc72Ex[0].Udata16[265].3 Acc72Ex[0].Udata16[265].4 Acc72Ex[0].Udata16[265].5 Acc72Ex[0].Udata16[265].6 Acc72Ex[0].Udata16[265].7 Acc72Ex[0].Udata16[265].8 Acc72Ex[0].Udata16[265].9 Acc72Ex[0].Udata16[266] Acc72Ex[0].Udata16[266].0 Acc72Ex[0].Udata16[266].1 Acc72Ex[0].Udata16[266].2 Acc72Ex[0].Udata16[266].3 Acc72Ex[0].Udata16[266].4 Acc72Ex[0].Udata16[266].5 Acc72Ex[0].Udata16[266].6 Acc72Ex[0].Udata16[266].7 Acc72Ex[0].Udata16[266].8 Acc72Ex[0].Udata16[266].9 Acc72Ex[0].Udata16[267] Acc72Ex[0].Udata16[267].2 Acc72Ex[0].Udata16[267].3 Acc72Ex[0].Udata16[267].4 Acc72Ex[0].Udata16[267].5 Acc72Ex[0].Udata16[267].6 Acc72Ex[0].Udata16[267].7 Acc72Ex[0].Udata16[267].8 Acc72Ex[0].Udata16[267].9 Acc72Ex[0].Udata16[388].0 Acc72Ex[0].Udata16[388].1 Acc72Ex[0].Udata16[388].2 Acc72Ex[0].Udata16[388].3 Acc72Ex[0].Udata16[388].4 Acc72Ex[0].Udata16[388].5 Acc72Ex[0].Udata16[388].6 Acc72Ex[0].Udata16[388].7 Acc72Ex[0].Udata16[388].8

Acc72Ex[0].Udata32[195]

Software setup

// CC0 CommunicationCOS	
#define CC0 RCX COMM COS READY	Acc72Ex[0].Udata16[392].0
#define CC0 RCX COMM COS RUN	
	Acc72Ex[0].Udata16[392].1
<pre>#define CC0_RCX_COMM_COS_BUS_ON #define CC0_RCX_COMM_COS_CONFIG_LOCKED #define CC0_RCX_COMM_COS_CONFIG_NEW #define_CC0_RCX_COMM_COS_RESTART_REQ</pre>	Acc72Ex[0].Udata16[392].2
#define CCU_RCX_COMM_COS_CONFIG_LOCKED	Acc72Ex[0].Udata16[392].3
#define CC0_RCX_COMM_COS_CONFIG_NEW	Acc72Ex[0].Udata16[392].4
#define CC0_RCX_COMM_COS_RESTART_REQ	Acc72Ex[0].Udata16[392].5
#define CC0_RCX_COMM_COS_RESTART_REQ_ENA	Acc72Ex[0].Udata16[392].6
#define CC0 RCX COMM COS DMA	Acc72Ex[0].Udata16[392].7
// CC0 Status Block	
#define CC0 ulCommunicationState	Acc72Ex[0].Udata32[197]
#define CC0_ulCommunicationError	Acc72Ex[0].Udata32[198]
#define CCO usVersion	Acc72Ex[0].Udata16[398]
#define CC0_usWatchdogTime	Acc72Ex[0].Udata16[399]
#define CCO_bPDInHskMode	Acc72Ex[0].Data8[800]
#define CC0_bPDInSource	Acc72Ex[0].Data8[801]
#define CC0_bPDOutHskMode	Acc72Ex[0].Data8[802]
#define CC0_bPDOutSource	Acc72Ex[0].Data8[803]
#define CC0_ulHostWatchdog	Acc72Ex[0].Udata32[201]
#define CCO_ulErrorCount	Acc72Ex[0].Udata32[202]
#define CC0 bErrorLogInd	Acc72Ex[0].Data8[812]
#define CC0 bErrorPDInCnt	Acc72Ex[0].Data8[813]
#define CC0 bErrorPDOutCnt	Acc72Ex[0].Data8[814]
#define CC0 bErrorSyncCnt	Acc72Ex[0].Data8[815]
#define CCO bSyncHskMode	Acc72Ex[0].Data8[816]
	Acc72Ex[0].Data8[817]
#define CC0_bSyncSource	ACC/2EA[V].Ddld0[01/]
// CC1 Control Block	
// CC1_Control Block	
#define CC1_RCX_APP_COS_APP_READY	Acc72Ex[0].Udata16[8196].0
#define CC1_RCX_APP_COS_BUS_ON	Acc72Ex[0].Udata16[8196].1
#define CC1_RCX_APP_COS_BUS_ON_ENABLE	Acc72Ex[0].Udata16[8196].2
#define CC1_RCX_APP_COS_INIT	Acc72Ex[0].Udata16[8196].3
#define CC1 RCX APP COS INIT ENABLE	Acc72Ex[0].Udata16[8196].4
#define CC1 RCX APP COS LOCK CFG	Acc72Ex[0].Udata16[8196].5
#define CC1 RCX APP COS LOCK CFG ENA	Acc72Ex[0].Udata16[8196].6
#define CC1 RCX APP COS DMA	Acc72Ex[0].Udata16[8196].7
#define CC1 RCX APP COS DMA ENABLE	Acc72Ex[0].Udata16[8196].8
#define CC1 ulDeviceWatchdog	Acc72Ex[0].Udata32[4099]
// CC1 CommunicationCOS	
#define CC1 RCX COMM COS READY	Acc72Ex[0].Udata16[8200].0
#define CC1 RCX COMM COS RUN	Acc72Ex[0].Udata16[8200].1
#define CC1_RCX_COMM_COS_BUS_ON	Acc72Ex[0].Udata16[8200].2
#define CC1_RCX_COMM_COS_CONFIG_LOCKED	Acc72Ex[0].Udata16[8200].3
#define CC1_RCX_COMM_COS_CONFIG_NEW	Acc72Ex[0].Udata16[8200].4
<pre>#define CC1_RCX_COMM_COS_RESTART_REQ</pre>	Acc72Ex[0].Udata16[8200].5
<pre>#define CC1_RCX_COMM_COS_RESTART_REQ_ENA</pre>	Acc72Ex[0].Udata16[8200].6
#define CC1_RCX_COMM_COS_DMA	Acc72Ex[0].Udata16[8200].7
// CC1_Status Block	
#define CC1 ulCommunicationState	Acc72Ex[0].Udata32[4101]
#define CC1 ulCommunicationError	Acc72Ex[0].Udata32[4102]
#define CC1 usVersion	Acc72Ex[0].Udata16[8206]
#define CC1_usWatchdogTime	Acc72Ex[0].Udata16[8207]
#define CC1 bPDInHskMode	Acc72Ex[0].Data8[16416]
#define CC1_brDinSkMode	Acc72Ex[0].Data8[16417]
#define CC1_bPD0ntSource #define CC1_bPD0utHskMode	ACC72Ex[0].Data8[16417] Acc72Ex[0].Data8[16418]
#define CC1_bPDOutSource	Acc72Ex[0].Data8[16419]
#define CC1_ulHostWatchdog	Acc72Ex[0].Udata32[4105]
#define CC1_ulErrorCount	Acc72Ex[0].Udata32[4106]
#define CC1_bErrorLogInd	Acc72Ex[0].Data8[16428]
#define CC1_bErrorPDInCnt	Acc72Ex[0].Data8[16429]
#define CC1_bErrorPDOutCnt	Acc72Ex[0].Data8[16430]
#define CC1_bErrorSyncCnt	Acc72Ex[0].Data8[16431]
#define CC1 bSyncHskMode	Acc72Ex[0].Data8[16432]
#define CC1_bSyncSource	Acc72Ex[0].Data8[16433]
// CC2 Control Block	
#define CC2 RCX APP COS APP READY	Acc72Ex[0].Udata16[16004].0
#define CC2 RCX APP COS BUS ON	Acc72Ex[0].Udata16[16004].1
"	

#define CC2_RCX_APP_COS_BUS_ON_ENABLE Acc72Ex[0].Udata16[16004].2 Acc72Ex[0].Udata16[16004].3 #define CC2 RCX APP COS INIT #define CC2_RCX_APP_COS_INIT_ENABLE
#define CC2_RCX_APP_COS_LOCK_CFG Acc72Ex[0].Udata16[16004].4 Acc72Ex[0].Udata16[16004].5 Acc72Ex[0].Udata16[16004].6 #define CC2 RCX APP COS LOCK CFG ENA Acc72Ex[0].Udata16[16004].7 #define CC2 RCX APP COS DMA #define CC2 RCX APP COS DMA ENABLE Acc72Ex[0].Udata16[16004].8 #define CC2 ulDeviceWatchdog Acc72Ex[0].Udata32[8003] // CC2 CommunicationCOS #define CC2 RCX COMM COS READY Acc72Ex[0].Udata16[16008].0 #define CC2 RCX COMM COS RUN Acc72Ex[0].Udata16[16008].1 #define CC2_RCX_COMM_COS_RONAcc72Ex[0].Udata16[16008].1#define CC2_RCX_COMM_COS_CONFIG_LOCKEDAcc72Ex[0].Udata16[16008].2#define CC2_RCX_COMM_COS_CONFIG_LOCKEDAcc72Ex[0].Udata16[16008].4#define CC2_RCX_COMM_COS_RESTART_REQAcc72Ex[0].Udata16[16008].5#define CC2_RCX_COMM_COS_RESTART_REQ_ENAAcc72Ex[0].Udata16[16008].6#define CC2_RCX_COMM_COS_DMAAcc72Ex[0].Udata16[16008].7 #define CC2_RCX_COMM_COS_BUS_ON // CC2 Status Block #define CC2 ulCommunicationState Acc72Ex[0].Udata32[8005] #define CC2_ulCommunicationError Acc72Ex[0].Udata32[8006] #define CC2 usVersion Acc72Ex[0].Udata16[16014] #define CC2_usWatchdogTime Acc72Ex[0].Udata16[16015] #define CC2_bPDInHskMode
#define CC2_bPDInSource Acc72Ex[0].Data8[32032] Acc72Ex[0].Data8[32033] #define CC2_bPDOutHskMode Acc72Ex[0].Data8[32034] #define CC2_bPDOutSource
#define CC2_ulHostWatchdog Acc72Ex[0].Data8[32035] Acc72Ex[0].Udata32[8009] #define CC2_ulErrorCount Acc72Ex[0].Udata32[8010] #define CC2_bErrorLogInd
#define CC2_bErrorPDInCnt Acc72Ex[0].Data8[32044] Acc72Ex[0].Data8[32045] #define CC2 bErrorPDOutCnt Acc72Ex[0].Data8[32046] #define CC2 bErrorSyncCnt Acc72Ex[0].Data8[32047] #define CC2_bSyncHskMode
#define CC2_bSyncSource Acc72Ex[0].Data8[32048] Acc72Ex[0].Data8[32049] // CC3 Control Block #define CC3 RCX APP COS APP READY #define CC3_RCX_APP_COS_BUS_ON #define CC3_RCX_APP_COS_BUS_ON Acc72Ex[0].Udata16[23812].0 Acc72Ex[0].Udata16[23812].1 ACC72Ex[0].Udata16[23812].2 #define CC3 RCX APP COS BUS ON ENABLE #define CC3_RCX_APP_COS_INIT
#define CC3_RCX_APP_COS_INIT_ENABLE Acc72Ex[0].Udata16[23812].3 Acc72Ex[0].Udata16[23812].4 #define CC3 RCX APP COS LOCK CFG Acc72Ex[0].Udata16[23812].5 Acc72Ex[0].Udata16[23812].6 Acc72Ex[0].Udata16[23812].7 #define CC3_RCX_APP_COS_LOCK_CFG_ENA
#define CC3_RCX_APP_COS_DMA #define CC3_RCX_APP_COS_DMA_ENABLE Acc72Ex[0].Udata16[23812].8 #define CC3 ulDeviceWatchdog Acc72Ex[0].Udata32[11907] // CC3 CommunicationCOS #define CC3 RCX COMM COS READY Acc72Ex[0].Udata16[23816].0 #define CC3_RCX_COMM_COS_RUN Acc72Ex[0].Udata16[23816].1 Acc72Ex[0].Udata16[23816].1 Acc72Ex[0].Udata16[23816].2 Acc72Ex[0].Udata16[23816].3 Acc72Ex[0].Udata16[23816].4 Acc72Ex[0].Udata16[23816].5 Acc72Ex[0].Udata16[23816].5 #define CC3_RCX_COMM_COS_BUS_ON
#define CC3_RCX_COMM_COS_CONFIG_LOCKED #define CC3_RCX_COMM_COS_CONFIG_NEW
#define CC3_RCX_COMM_COS_RESTART_REQ
#define CC3_RCX_COMM_COS_RESTART_REQ_ENA #define CC3 RCX COMM COS DMA Acc72Ex[0].Udata16[23816].7 // CC3 Status Block #define CC3_ulCommunicationState Acc72Ex[0].Udata32[11909] #define CC3 ulCommunicationError Acc72Ex[0].Udata32[11910] #define CC3_usVersion
#define CC3_usWatchdogTime Acc72Ex[0].Udata16[23822] Acc72Ex[0].Udata16[23823] #define CC3 bPDInHskMode Acc72Ex[0].Data8[47648] #define CC3 bPDInSource Acc72Ex[0].Data8[47649] #define CC3_bPDOutHskMode
#define CC3_bPDOutSource Acc72Ex[0].Data8[47650] Acc72Ex[0].Data8[47651] #define CC3 ulHostWatchdog Acc72Ex[0].Udata32[11913] #define CC3_ulErrorCount
#define CC3_bErrorLogInd Acc72Ex[0].Udata32[11914] Acc72Ex[0].Data8[47660]

<pre>#define CC3_bErrorPDInCnt #define CC3_bErrorPDOutCnt #define CC3_bErrorSyncCnt #define CC3_bSyncHskMode #define CC3_bSyncSource</pre>		Acc72Ex[0].Data8[47661] Acc72Ex[0].Data8[47662] Acc72Ex[0].Data8[47663] Acc72Ex[0].Data8[47664] Acc72Ex[0].Data8[47665]
<pre>#ifdefPROFIBUS_DP_Master_ #define CC0_PD0_OUT_OFFSET_2BYTE #define CC0_PD0_OUT_SIZE_2BYTE #define CC0_PD0_IN_OFFSET_2BYTE #define CC0_PD0_IN_SIZE_2BYTE #define CC0_PD1_OUT_OFFSET_2BYTE #define CC0_PD1_OUT_SIZE_2BYTE #define CC0_PD1_IN_OFFSET_2BYTE #define CC0_PD1_IN_OFFSET_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #endif</pre>	\$980 2880 \$14C0 2880 \$8C0 32 \$8E0 32	
<pre>#ifdefPROFIBUS_DP_Slave_ #define CC0_PD0_OUT_OFFSET_2BYTE #define CC0_PD0_OUT_SIZE_2BYTE #define CC0_PD0_IN_OFFSET_2BYTE #define CC0_PD0_IN_SIZE_2BYTE #define CC0_PD1_OUT_OFFSET_2BYTE #define CC0_PD1_OUT_SIZE_2BYTE #define CC0_PD1_IN_OFFSET_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #endif</pre>	\$980 768 \$C80 768 \$8C0 32 \$8E0 32	
<pre>#ifdefDeviceNet_Master #define CC0_PD0_OUT_OFFSET_2BYTE #define CC0_PD0_OUT_SIZE_2BYTE #define CC0_PD0_IN_OFFSET_2BYTE #define CC0_PD0_IN_SIZE_2BYTE #define CC0_PD1_OUT_OFFSET_2BYTE #define CC0_PD1_OUT_SIZE_2BYTE #define CC0_PD1_IN_OFFSET_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #endif</pre>	\$980 2880 \$14C0 2880 \$8C0 32 \$8E0 32	
<pre>#ifdefDeviceNet_Slave #define CC0_PD0_OUT_OFFSET_2BYTE #define CC0_PD0_OUT_SIZE_2BYTE #define CC0_PD0_IN_OFFSET_2BYTE #define CC0_PD0_IN_SIZE_2BYTE #define CC0_PD1_OUT_OFFSET_2BYTE #define CC0_PD1_OUT_SIZE_2BYTE #define CC0_PD1_IN_OFFSET_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #endif</pre>	\$980 768 \$C80 768 \$8C0 32 \$8E0 32	
<pre>#ifdefCANopen_Master #define CC0_PD0_OUT_OFFSET_2BYTE #define CC0_PD0_OUT_SIZE_2BYTE #define CC0_PD0_IN_OFFSET_2BYTE #define CC0_PD1_OUT_OFFSET_2BYTE #define CC0_PD1_OUT_OFFSET_2BYTE #define CC0_PD1_OUT_SIZE_2BYTE #define CC0_PD1_IN_OFFSET_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #endif</pre>	\$980 2880 \$14C0 2880 \$8C0 32 \$8E0 32	
<pre>#ifdefCANopen_Slave #define CC0_PD0_OUT_OFFSET_2BYTE #define CC0_PD0_OUT_SIZE_2BYTE #define CC0_PD0_IN_OFFSET_2BYTE #define CC0_PD0_IN_SIZE_2BYTE #define CC0_PD1_OUT_OFFSET_2BYTE #define CC0_PD1_OUT_SIZE_2BYTE #define CC0_PD1_IN_OFFSET_2BYTE #define CC0_PD1_IN_SIZE_2BYTE</pre>	\$980 768 \$C80 768 \$8C0 32 \$8E0 32	

#endif

#ifdef	CC_Link_Slave	
#define	CC0 PD0 OUT OFFSET 2BYTE	\$980
	CC0_PD0_OUT_SIZE_2BYTE	768
	CCO PDO IN OFFSET 2BYTE	\$C80
	CC0 PD0 IN SIZE 2BYTE	768
#define	CC0 PD1 OUT OFFSET 2BYTE	\$8C0
#define	CC0 PD1 OUT SIZE 2BYTE	32
	CC0 PD1 IN OFFSET 2BYTE	\$8E0
		32
	CC0_PD1_IN_SIZE_2BYTE	32
#endif		
#ifdof	EthorCAM Mostor	
	EtherCAT_Master	¢000
#derine	CC0_PD0_OUT_OFFSET_2BYTE	\$980
#derine	CC0_PD0_OUT_SIZE_2BYTE CC0_PD0_IN_OFFSET_2BYTE	2880
#define	CCU_PDU_IN_OFFSET_ZBYTE	\$14C0
	CC0_PD0_IN_SIZE_2BYTE	2880
#define	CC0_PD1_OUT_OFFSET_2BYTE	\$8C0
#define	CCO_PD1_OUT_SIZE_2BYTE CCO_PD1_IN_OFFSET_2BYTE	32
#define	CC0_PD1_IN_OFFSET_2BYTE	\$8E0
	CC0_PD1_IN_SIZE_2BYTE	32
#endif		
#ifdef	EtherCAT_Slave	* * * *
#define	CC0_PD0_OUT_OFFSET_2BYTE	\$980
#define	CCO PDO OUT SIZE 2BYTE	2880
#define	CC0_PD0_IN_OFFSET_2BYTE	\$14C0
#define	CC0_PD0_IN_SIZE_2BYTE	2880
#define	CC0 PD1 OUT OFFSET 2BYTE	\$8C0
#define	CC0_PD1_OUT_SIZE_2BYTE	32
#define	CC0_PD1_IN_OFFSET_2BYTE	\$8E0
#define	CC0 PD1 IN SIZE 2BYTE	32
#endif		
#ifdef	EtherNetIP_Scanner_Master	
#define	CC0_PD0_OUT_OFFSET_2BYTE	\$980
#define	CC0_PD0_OUT_OFFSET_2BYTE CC0_PD0_OUT_SIZE_2BYTE	2880
#define	CC0_PD0_IN_OFFSET_2BYTE CC0_PD0_IN_SIZE_2BYTE	\$14C0
#define	CCO PDO IN SIZE 2BYTE	2880
#define	CCO PD1 OUT OFFSET 2BYTE	\$8C0
#define	CC0_PD1_OUT_SIZE_2BYTE	32
#define	CC0 PD1 IN OFFSET 2BYTE	\$8E0
	CC0 PD1 IN SIZE 2BYTE	32
#endif		
#ifdef	EtherNetIP_Adapter_Slave	
#define	CC0_PD0_OUT_OFFSET_2BYTE	\$980
#define	CC0 PD0 OUT SIZE 2BYTE	2880
#define	CC0_PD0_OUT_SIZE_2BYTE CC0_PD0_IN_OFFSET_2BYTE	\$14C0
#define	CC0 PD0 IN SIZE 2BYTE	2880
	CC0 PD1 OUT OFFSET 2BYTE	\$8C0
	CC0 PD1 OUT SIZE 2BYTE	32
#define	CC0 PD1 IN OFFSET 2BYTE	\$8E0
#define	CCO PD1 IN SIZE 2BYTE	32
#endif		52
#ifdef	Open Modbus TCP	
	CC0 PD0 OUT OFFSET 2BYTE	\$980
	CC0 PD0 OUT SIZE 2BYTE	2880
	CC0 PD0 IN OFFSET 2BYTE	\$14C0
	CC0_PD0_IN_SIZE_2BYTE	2880
	CC0 PD1 OUT OFFSET 2BYTE	\$8C0
#define	CC0 PD1 OUT SIZE 2BYTE	32
	CC0 PD1 IN OFFSET 2BYTE	\$8E0
	CC0 PD1 IN SIZE 2BYTE	32
#endif	000_1D1_1N_0125_2D115	54
" CIIULL		
#ifdof		
#ILGEL	PROFINET IO Controller Master	
	PROFINET_IO_Controller_Master CC0 PD0 OUT OFFSET 2BYTE	\$980
#define	PROFINET_IO_Controller_Master CC0_PD0_OUT_OFFSET_2BYTE CC0_PD0_OUT_SIZE_2BYTE	

<pre>#define CC0_PD0_IN_OFFSET_2BYTE #define CC0_PD0_IN_SIZE_2BYTE #define CC0_PD1_OUT_OFFSET_2BYTE #define CC0_PD1_OUT_SIZE_2BYTE #define CC0_PD1_IN_OFFSET_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #endif</pre>	32
<pre>#ifdefPROFINET_IO_Device_Slave #define CC0_PD0_OUT_OFFSET_2BYTE #define CC0_PD0_OUT_SIZE_2BYTE #define CC0_PD0_IN_OFFSET_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #define CC0_PD1_OUT_OFFSET_2BYTE #define CC0_PD1_OUT_SIZE_2BYTE #define CC0_PD1_IN_OFFSET_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #define CC0_PD1_IN_SIZE_2BYTE #endif</pre>	2880 \$14C0 2880 \$8C0 32
ptr CC0_PD0_OUT16(CC0_PD0_OUT_SIZE_2 ptr CC0_PD0_IN16(CC0_PD0_IN_SIZE_2By ptr CC0_PD1_OUT16(CC0_PD1_OUT_SIZE_2 ptr CC0_PD1_IN16(CC0_PD1_IN_SIZE_2By	(TE) ->*; 2BYTE) ->*;

Initialization PLC

Recall that ACC-72EX requires a reset after each power up, power cycle, \$\$\$ (reset), or \$\$\$*** (factory default reset). This can be achieved with a startup (or initialization) PLC. Example:

```
// ACC-72EX initialization PLC
open plc Acc72EX StartupPLC
local endtime;
                                               // Disable all other tasks
disable plc 2..31
// Defining pointers for system channel mailboxes
L_{0}=0
while (LO<84)
{
     CMD"SSMB Data8(%d)->Acc72Ex[0].Data8[%d]",L0,L0+300
     sendallcmds
     L0++
}
L0=0
while (LO<42)
{
     CMD"SSMB Data16(%d) ->Acc72Ex[0].uData16[%d]",L0,L0+150
     sendallcmds
     L0++
}
L0=0
while (LO<21)
{
     CMD"SSMB Data32(%d)->Acc72Ex[0].uData32[%d]",L0,L0+75
     sendallcmds
     L0++
ļ
L0=0
while(LO<84)
{
     CMD"SRMB Data8(%d)->Acc72Ex[0].Data8[%d]",L0,L0+428
     sendallcmds
     L0++
}
L0=0
while (LO<42)
{
     CMD"SRMB Data16(%d) ->Acc72Ex[0].uData16[%d]", L0, L0+214
     sendallcmds
```

```
L0++
}
L_{0}=0
while(L0<21)</pre>
{
      CMD"SRMB Data32(%d)->Acc72Ex[0].uData32[%d]",L0,L0+107
      sendallcmds
      L0++
}
// Defining pointers to Out/In PDOs
L0=0
while (LO<CCO PDO OUT SIZE 2BYTE)
{
      CMD"CC0 PD0 OUT16(%d)-
>Acc72Ex[0].uData16[%d]",L0,L0+CC0 PD0 OUT OFFSET 2BYTE;
      sendallcmds
      L0++
}
L0=0
while (LO< CCO PDO IN SIZE 2BYTE)
{
      CMD"CC0 PD0 IN16(%d)-
>Acc72Ex[0].uData16[%d]",L0,L0+CC0 PD0 IN OFFSET 2BYTE
      sendallcmds
      L0++
}
L0=0
while (LO<CCO PD1 OUT SIZE 2BYTE)
{
      CMD"CC0 PD1 OUT16(%d)-
>Acc72Ex[0].uData16[%d]",L0,L0+CC0 PD1 OUT OFFSET 2BYTE
     sendallcmds
     L0++
}
L0=0
while (LO< CCO PD1 IN SIZE 2BYTE)
{
      CMD"CC0 PD1 IN16(%d) ->Acc72Ex[0].uData16[%d]",L0,L0+
CC0 PD1 IN OFFSET 2BYTE
     sendallcmds
      L0++
}
SCtrl_ulSystemCommandCOS=$55AA55AA // Reset token for MASTER Unit
HCSC HSF RESET=1
                                    // Reset bit, token required for reset to
complete
CommErrorFlag=0;
                                           // Reset Time-out Timer
endtime = Sys.Time + 2;
while (CommErrorFlag==0 && HCSC NSF READY==0) // Wait for reset to complete
{
      if (endtime<Sys.Time)</pre>
                                                // Check for reset timeout
      {
            CommErrorFlag = 1;
      }
call Timer(0.100); // 100 msec
```

```
// Toggle Communication Channel 0's Change of State Acknowledge bit in
// order to read the CCO RCX COMM COS RUN which is a part of Communication
// Channel 0 State Register
HCCC0 HCF NETX COS ACK = HCCC0 HCF_NETX_COS_ACK ^ 1
if (CommErrorFlag==0)
{
       while ((CC0 RCX COMM COS RUN) == 0) // Wait for comm tasks to
       {
              // start on COMX modules
              // Repeating the toggle action for next while loop read
              HCCCO HCF NETX COS ACK = HCCCO HCF NETX COS ACK ^ 1
              call Timer(0.010); // 10 msec
      enable plc Acc72EX_WatchdogPLC // Enable the Watchdog plc
enable plc Acc72EX_PDO_WritePLC // Enable the write plc
enable plc Acc72EX_PDO_ReadPLC // Enable the write plc
       }
disable plc Acc72EX StartupPLC
close
```

The above PLC uses a Timer subprogram call that must be added to the PMAC Script Language \rightarrow Libraries folder of the IDE project:

```
open subprog Timer(wait_duration) // wait_duration in seconds
local EndTime = Sys.Time + wait_duration;
while(Sys.Time < EndTime){}
close</pre>
```

Startup

To enable this startup PLC at power-up or reset, add the following line to pp_startup.txt in the Configuration folder:

enable plc Acc72EX StartupPLC

Watchdog Function

The host Watchdog and the device Watchdog cells in the control block of each of the communication channels allow the operating system running on the netX to supervise the host or UMAC application and vice versa. There is no Watchdog function for the system block or for the handshake channel. The Watchdog for the channels is located in the control block of the status block of each communication channel.

The netX firmware reads the content of the device Watchdog cell, increments the value by one and copies it back into the host Watchdog location. Then, the application has to copy the new value from the host Watchdog location into the device Watchdog location. Copying the host Watchdog cell to the device Watchdog cell has to happen in the configured Watchdog time. When the overflow occurs, the firmware starts over and a one appears in the host Watchdog cell. A zero turns off the Watchdog and therefore never appears in the host Watchdog cell in the regular process.

The minimum Watchdog time is 20 ms. The application can start the Watchdog function by copying any value unequal to zero into device Watchdog cell. A zero in the device Watchdog location stops the Watchdog function. The Watchdog timeout is configurable in SYCON.net and downloaded to the netX firmware.

If the application fails to copy the value from the host Watchdog location to the device Watchdog location within the configured Watchdog time, the protocol stack will interrupt all network connections immediately, regardless of their current state. If the Watchdog tripped, then power cycling, channel reset, or channel initialization will allow the communication channel to open network connections again.

Here is sample code for copying the host Watchdog location to the device Watchdog location:

Enabling the Communication Bus

Using the Bus On flag (CCx_RCX_APP_COS_BUS_ON, where x is the communication channel number), the host or UMAC application allows or disallows the netX firmware to open network connections. This flag is used together with the Bus On Enable flag

(CCx_RCX_APP_COS_BUS_ON_ENABLE, where x is the communication channel number). If set, the netX firmware tries to open network connections; if cleared, no connections are allowed, and open connections are closed. If the Bus On Enable flag is set, it enables the execution of the Bus On command in the netX firmware.

```
CC0_RCX_APP_COS_BUS_ON=1// Setting the Bus On flag for 1st ACC-72EXCC0_RCX_APP_COS_BUS_ON_ENABLE=1// Enabling the execution of Bus On Flag for 1st ACC-72EXS_CC0_RCX_APP_COS_BUS_ON=1// Setting the Bus On flag for 2nd ACC-72EXS_CC0_RCX_APP_COS_BUS_ON_ENABLE=1// Enabling the execution of Bus On Flag for 2nd ACC-72EX
```

Locating the Input/Output Data Image in PMAC

The header file provided for use with ACC-72EX provides proper addressing and offsets for each of the PDOs available for each communication module. There are also pointers declared in the header file and are defined as a part of the initialization PLC shown above. These pointers will be used to access different PDOs defined by SYCON.net software.

The following example PLCs are for reference only in order to demonstrate the proper handshaking necessary for reading and writing data to ACC-72EX from Power PMAC.

```
/****
// ACC-72EX Writing to PDO Sample PLC
open plc Acc72EX PDO WritePLC
if (HCCC0 HCF PD0 OUT CMD == HCCC0 NCF PD0 OUT ACK)
                                          {
// Making sure the ACK flag matches the CMD
// flag before writing the value to the
// output data image register
    P200=P200+1
    CC0 PD0 OUT16(0) = P200;
                        // write the output data image register
    // Toggle the CMD flag (^: XOR)
    HCCC0 HCF PD0 OUT CMD = HCCC0 HCF PD0 OUT CMD^1
    // indicating write completion
}
close
```

DIAGNOSTICS

LEDs

There is one system LED (SYS LED) per ACC-72EX. SYS LED is always present as described below. There are up to 4 LEDs per communication and application channel. These LEDs, like the communication channel LED (COM LED), are network-specific and are described separately.

SYS LED

The system status LED (SYS LED) is always available. It indicates the state of the system and its protocol stacks. The following blink patterns are defined:

Color	State	Meaning
Yellow	Flashing Cyclically at 1 Hz	netX is in Boot Loader Mode and is Waiting for Firmware
		Download
	Solid	netX is in Boot Loader Mode, but an Error Occurred
Green	Solid	netX Operating System is Running and a Firmware is Started
Vellow /	Flashing Alternating	2nd Stage Bootloader is active
Green		
Off	N/A	netX has no Power Supply or Hardware Defect Detected

PROFIBUS-DP – Master – OPT10

The following table describes the meaning of the LEDs for the comX PROFIBUS-DP Master communication modules (COMX 100CA-DP/ COMX100CN-DP) when the firmware of the PROFIBUS DP Master protocol is loaded to the comX communication module:

COM LED (COM0)

Color	State	Meaning
Green	Flashing acyclic	No configuration or stack error
Green	Flashing cyclic	Profibus is configured, but bus communication is not yet released from the application
Green	On	Communication to all Slaves is established
Red	Flashing cyclic	Communication to at least one Slave is disconnected
Red	On	Communication to one/all Slaves is disconnected

PROFIBUS-DP – Slave – OPT11

The subsequent table describes the meaning of the LEDs for the comX PROFIBUS-DP Slave communication modules (COMX CA-DP/ COMX CNDP) when the firmware of the PROFIBUS DP Slave protocol is loaded to the comX communication module.

COM LED (COM0)

Color	State	Meaning
Green	On	RUN, cyclic communication
Red	Flashing cyclic	STOP, no communication, connection error
Red	Flashing acyclic	not configured

DeviceNet – Master – OPT20

The following table describes the meaning of the LEDs for the comX communication modules when the firmware of the DeviceNet Master protocol is loaded to the comX communication module:

MNS LED (COM0)

Color	State	Meaning
Green	On	Device is online and has established one or more
		connections
Green	Flashing	Device is online and has established no connection
Green/Red	Green/Red/Off	Self-test after power on:
		Green on for 0,25 s, then red on for 0,25 s, then off
Red	Flashing	Connection timeout
Red	On	Critical connection failure; device has detected a network error: duplicate MAC-ID
		or severe error in CAN network (CAN-bus off)
Red	Off	After start of the device and during duplicate MAC-ID check

DeviceNet – Slave – OPT21

The following table describes the meaning of the LEDs for the comX communication modules when the firmware of the DeviceNet Slave protocol is loaded to the comX communication module:

MNS LED (COM0)

Color	State	Meaning
Green	On	Device is online and has established one or more
		connections
Green	Flashing	Device is online and has established no connection
Green/Red	Green/Red/Off	Self-test after power on:
		Green on for 0,25 s, then red on for 0,25 s, then off
Red	Flashing	Connection timeout
Red	On	Critical connection failure; device has detected a network error: duplicate MAC-ID
		or severe error in CAN network (CAN-bus off)
Red	Off	After start of the device and during duplicate MAC-ID check

CANopen – Master – OPT30

The following table describes the meaning of the LEDs for the comX CANopen Master communication modules (COMX-CA-CO/ COMX-CNCOM) when the firmware of the CANopen Master protocol is loaded to the comX communication module:

CAN LED (COM0)

Color	State	Meaning
Green	Off	The device is executing a reset
Green	Single Flash	STOPPED: The Device is in STOPPED state
Green	Blinking	PREOPERATIONAL: The Device is in the PREOPERATIONAL state
		The indicator turns on and off with a frequency of 2,5 Hz: on for 200 ms, followed by off for 200 ms.
Green	On	OPERATIONAL: The Device is in the OPERATIONAL state
Red	Single flash	Warning Limit reached: At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames). The indicator shows one short flash (200 ms) followed by a long off phase (1,000 ms).
Red	Double flash	Error Control Event: A guard event (NMT Slave or NMTmaster) or a heartbeat event (Heartbeat consumer) has occurred. The indicator shows a sequence of two short flashes (each 200 ms), separated by a short off phase (200 ms). The sequence is finished by a long off phase (1,000 ms).
Red	On	Bus Off: The CAN controller is bus off

CANopen – Slave – OPT31

The following table describes the meaning of the LEDs for the comX CANopen Slave communication modules (COMX-CA-CO/ COMX-CNCOS) when the firmware of the CANopen Slave protocol is loaded to the comX communication module:

CAN LED (COM0)

Color	State	Meaning
Green	Off	The device is executing a reset
Green	Single Flash	STOPPED: The Device is in STOPPED state
Green	Blinking	PREOPERATIONAL: The Device is in the PREOPERATIONAL state
		The indicator turns on and off with a frequency of 2,5 Hz: on for 200 ms, followed
		by off for 200 ms.
Green	On	OPERATIONAL: The Device is in the OPERATIONAL state
Red	Off	No Error: The Device is in working condition
Red	Single flash	Warning Limit reached: At least one of the error counters of the CAN controller
		has reached or exceeded the warning level (too many error frames).
		The indicator shows one short flash (200 ms) followed by a long off phase (1,000
		ms).
Red	Double flash	Error Control Event: A guard event (NMT Slave or NMTmaster) or a heartbeat
		event (Heartbeat consumer) has occurred.
		The indicator shows a sequence of two short flashes (each 200 ms), separated by
		a short off phase (200 ms). The sequence is finished by a long off phase (1,000
		ms).
Red	On	Bus Off: The CAN controller is bus off

CC-Link – Slave – OPT51

The following table describes the meaning of the LEDs for the comX CCLink Slave communication modules (COMX 100CA-CCS/ COMX 100CNCCS) when the firmware of the CC-Link Slave protocol is loaded to the comX communication module:

RUN/ERR LED (COM0)

Color	State	Meaning
Green	Off	1. Before participating in the network
		2. Unable to detect carrier
		3. Timeout
		4. Resetting hardware
Green	On	Receive both refresh and polling signals or just the refresh signal normally, after
		participating in the network.
Red	Off	1. Normal communication
		2. Resetting hardware
Red	Blinking	The switch setting has been changed from the setting at the reset cancellation
		(blinks for 0.4 sec.).
Red	On	1. CRC error
		2. Address parameter error (0, 65 or greater is set including the number of
		occupied stations)
		3. Baud rate switch setting error during cancellation of reset (5 or greater)

EtherCAT – Master – OPT60

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX 100CA-RE/ COMX 100CN-RE) when the firmware of the EtherCAT Master protocol is loaded to the comX communication module:

RUN LED (COM0)

Color	State	Meaning
Green	Off	INIT: The device is in state INIT
Green	Blinking	PRE-OPERATIONAL: The device is in PREOPERATIONAL state
Green	Flickering	BOOT: Device is in BOOT state
Green	Single Flash	SAFE-OPERATIONAL: The device is in SAFE-OPERATIONAL state
Green	On	OPERATIONAL: The device is in OPERATIONAL state

ERR LED (COM1)

Color	State	Meaning
Red	Off	Master has no errors
Red	On	Master has detected a communication error. The error is indicated in the DPM

LINK LED

Green LED on ETH0 connector

Color	State	Meaning
Green	On	A link is established
Green	Off	No link established

ACT LED

Yellow LED on ETH0 connector

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

LED State Definition for EtherCAT Master for the RUN and ERR LEDs

Color	Meaning
On	The indicator is constantly on.
Off	The indicator is constantly off.
Blinking	The indicator turns on and off with a frequency of 2,5 Hz: on for 200 ms, followed by off for 200 ms.
Flickering	The indicator turns on and off with a frequency of approximately 10 Hz: on for approximately 50 ms, followed by
	off for 50 ms.
Single Flash	The indicator shows one short flash (200 ms) followed by a long off phase (1,000 ms).
Double Flash	The indicator shows a sequence of two short flashes (each 200 ms), separated by a short off phase (200 ms). The
	sequence is finished by a long off phase (1,000 ms).

EtherCAT – Slave – OPT61

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX 100CA-RE/ COMX 100CN-RE) when the firmware of the EtherCAT Slave protocol is loaded to the comX communication module:

RUN LED (COM0)

Color	State	Meaning
Green	Off	INIT: The device is in state INIT
Green	Blinking	PRE-OPERATIONAL: The device is in PREOPERATIONAL state
Green	Flickering	BOOT: Device is in BOOT state
Green	Single Flash	SAFE-OPERATIONAL: The device is in SAFE-OPERATIONAL state
Green	On	OPERATIONAL: The device is in OPERATIONAL state

ERR LED (COM1)

Color	State	Meaning
Red	Off	No error: The EtherCAT communication of the device is in working condition
Red	Blinking	Invalid Configuration: General Configuration Error Possible reason: State change commanded by master is impossible due to register or object settings.
Red	Single Flash	Local Error: Slave device application has changed the EtherCAT state autonomously. Possible reason 1: A host Watchdog timeout has occurred. Possible reason 2: Synchronization Error, device enters Safe-Operational automatically.
Red	Double Flash	Application Watchdog Timeout: An application Watchdog timeout has occurred. Possible reason: Sync Manager Watchdog timeout.

LINK/ACT LED

Green LED on ETH0(IN) / ETH1(OUT) connectors:

Color	State	Meaning
Green	On	A link is established
Green	Flashing	The device sends/receives Ethernet frames
Green	Off	No link established

Yellow LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Yellow	-	-

LED State Definition for EtherCAT Slave for the RUN and ERR LEDs

Color	Meaning
On	The indicator is constantly on.
Off	The indicator is constantly off.
Blinking	The indicator turns on and off with a frequency of 2,5 Hz: on for 200 ms, followed by off for 200 ms.
Single Flash	The indicator shows one short flash (200 ms) followed by a long off phase (1,000 ms).
Double Flash	The indicator shows a sequence of two short flashes (each 200 ms), separated by a short off phase (200 ms). The
	sequence is finished by a long off phase (1,000 ms).

EtherNet/IP – Scanner/Master – OPT70

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX-CA-RE/ COMX-CNRE) when the firmware of the EtherNet/IP Scanner (Master) protocol is loaded to the comX communication module:

MS LED (COM0)

Color	State	Meaning
Green	On	Device operational: If the device is operating correctly, the module status
		indicator shall be steady green.
Green	Flashing	Standby: If the device has not been configured, the module status indicator shall
		be flashing green.
Red	On	Major fault: If the device has detected a non-recoverable major fault, the module
		status indicator shall be steady red.
Green	Flashing	Minor fault: If the device has detected a recoverable minor fault, the module
		status indicator shall be flashing red. NOTE: An incorrect or inconsistent
		configuration would be considered a minor fault.
Red/Green	Flashing	Self-test: While the device is performing its power up testing, the module status
		indicator shall be flashing green/red.
-	Off	No power: If no power is supplied to the device, the module status indicator shall
		be steady off.

NS LED (COM1)

Color	State	Meaning
Green	On	Connected: If the device has at least one established connection (even to the
		Message Router), the network status indicator shall be steady green.
Green	Flashing	No connections: If the device has no established connections, but has obtained an
		IP address, the network status indicator shall be flashing green.
Red	On	Duplicate IP: If the device has detected that its IP address is already in use, the
		network status indicator shall be steady red.
Red	Flashing	Connection timeout: If one or more of the connections in which this device is the
		target has timed out, the network status indicator shall be flashing red. This shall
		be left only if all timed out connections are reestablished or if the device is reset.
Red/Green	Flashing	Self-test: While the device is performing its power up testing, the network status
		indicator shall be flashing green/red.
-	Off	Not powered, no IP address: If the device does not have an IP address (or is
		powered off), the network status indicator shall be steady off.

LINK LED

Green LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

EtherNet/IP – Adaptor/Slave – OPT71

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX-CA-RE/ COMX-CNRE) when the firmware of the EtherNet/IP Adapter (Slave) protocol is loaded to the comX communication module:

MS LED (COM0)

Color	State	Meaning
Green	On	Device operational: If the device is operating correctly, the module status indicator shall be steady green.
Green	Flashing	Standby: If the device has not been configured, the module status indicator shall be flashing green.
Red	On	Major fault: If the device has detected a non-recoverable major fault, the module status indicator shall be steady red.
Green	Flashing	Minor fault: If the device has detected a recoverable minor fault, the module status indicator shall be flashing red. NOTE: An incorrect or inconsistent configuration would be considered a minor fault.
Red/Green	Flashing	Self-test: While the device is performing its power up testing, the module status indicator shall be flashing green/red.
-	Off	No power: If no power is supplied to the device, the module status indicator shall be steady off.

NS LED (COM1)

Color	State	Meaning
Green	On	Connected: If the device has at least one established connection (even to the
		Message Router), the network status indicator shall be steady green.
Green	Flashing	No connections: If the device has no established connections, but has obtained an
		IP address, the network status indicator shall be flashing green.
Red	On	Duplicate IP: If the device has detected that its IP address is already in use, the
		network status indicator shall be steady red.
Red	Flashing	Connection timeout: If one or more of the connections in which this device is the
		target has timed out, the network status indicator shall be flashing red. This shall
		be left only if all timed out connections are reestablished or if the device is reset.
Red/Green	Flashing	Self-test: While the device is performing its power up testing, the network status
		indicator shall be flashing green/red.
-	Off	Not powered, no IP address: If the device does not have an IP address (or is
		powered off), the network status indicator shall be steady off.

LINK LED

Green LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

Open Modbus/TCP – OPT80

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX 100CA-RE/ COMX 100CN-RE) when the firmware of the Open Modbus/TCP protocol is loaded to the comX communication module:

RUN LED (COM0)

Color	State	Meaning
Green	Off	Not Ready
		OMB task is not ready
Green	Flashing cyclic with 1Hz	Ready, not configured yet
		OMB task is ready and not configured yet
Green	Flashing cyclic with 5Hz	Waiting for Communication:
		OMB task is configured
Green	On	Connected:
		OMB task has communication – at least one TCP connection is established

ERR LED (COM1)

Color	State	Meaning
Red	Off	No communication error
Red	Flashing cyclic with 2Hz (On/Off ratio 25%)	System error
Red/Green	On	Communication error active

LINK LED

Green LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

PROFINET IO – Controller – OPT90

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX 100CA-RE/ COMX 100CN-RE) when the firmware of the PROFINET IO-RT Controller protocol is loaded to the comX communication module:

SF LED (COM0)

Color	State	Meaning
Red	On	(together with BF "red ON") No valid Master license
Red	Flashing cyclic with 2Hz	System error: Invalid configuration, Watchdog error or internal error
Red	Off	No error

BF LED (COM1)

Color	State	Meaning
Red	On	No Connection: No Link or (together with SF "red ON")
		No valid Master license
Red	Flashing cyclic with 2Hz	Configuration fault: not all configured IO-Devices are connected.
Red	Off	No error

LINK LED

Green LED on ETH0 / ETH1 connectors:

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors:

ſ	Color	State	Meaning
	Yellow	Flashing	The device sends/receives Ethernet frames

PROFINET IO – Device – OPT91

The following table describes the meaning of the LEDs for the comX Real-Time Ethernet communication modules (COMX-CA-RE/ COMX-CNRE) when the firmware of the PROFINET IO-RT-Device protocol is loaded to the comX communication module:

SF LED (COM0)

Color	State	Meaning
Red	On	Watchdog timeout; channel, generic or extended diagnosis present; system error
Red	Flashing cyclic with 2Hz (for 3 seconds)	DCP signal service is initiated via the bus
Red	Off	No error

BF LED (COM1)

Color	State	Meaning
Red	On	No configuration; or low speed physical link; or no physical link
Red	Flashing cyclic with 2Hz	No data exchange
Red	Off	No error

LINK LED

Green LED on ETH0 / ETH1 connectors

Color	State	Meaning
Green	On	A connection to the Ethernet exists
Green	Off	The device has no connection to the Ethernet

ACT LED

Yellow LED on ETH0 / ETH1 connectors

Color	State	Meaning
Yellow	Flashing	The device sends/receives Ethernet frames

APPENDIX A – SETUP EXAMPLES

SYCON.net Setup

The following is a sample setup using an ACC-72EX Ethernet IP slave with an Allen-Bradley CompactLogix controller (1769-L18ERM-BB1B) as a master. SYCON.net for netX 1.310 was used in this example.

With the power off, plug the ACC-72EX into the UBUS backplane, and then power the UMAC rack. Connect the diagnostic port to a USB port on the PC using a micro-USB type cable.

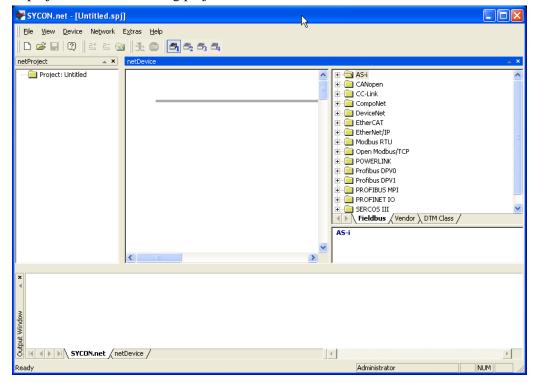
Launch the SYCON.NET software on the PC.



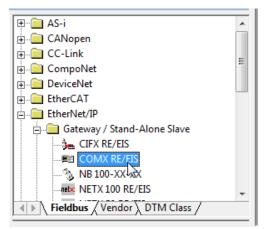
Enter the password:



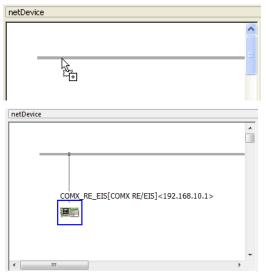
Start a new project or load an existing project from the File menu:



Select the COMX module, to which the USB is connected, from the Fieldbus protocol list:



Drag and drop the module onto the BusLine in the netDevice window (notice that the module can only be inserted on the BusLine):



Establish USB communications to the COMX gateway by right clicking on the device icon and selecting "Configuration...":

netDevice		
		^
COI	MX_RE_EIS[COMX RE/EIS]<192.168	.10.1>
<u>@</u>	Connect	
	Disconnect	
	Download	+
∢ [Upload	4
	Cut Copy	
	Paste	
	Configuration	
	Measured Value	
	Simulation Diagnostic	
	Additional Functions	
	Delete	
	Symbolic Name	

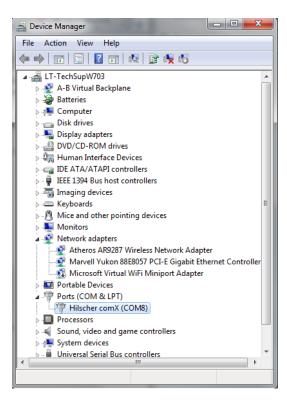
In the netDevice Configuration window, select the Driver folder under Settings folder in the Navigation Area, check the checkmark box for netX Driver on the driver list, and click Apply:

k netDevice - Configuration CO	MX_RE_EIS[COMX RE/EIS]<192.168.10.1>			
	RE/EIS r GmbH	Device Vendor		FD
Navigation Area		Driver		
 Settings Driver netX Driver Device Assignment Firmware Download Configuration General Electronic Keying 	Driver Stateway Driver for netX (V3.x) netX Driver	Version 0.9.1.2 1.101.1.5347	•	9-8E4D-10986A68EA91} 5-8405-6E12FC88EE62)
Connection Assembly Device Settings Description Device Info				

Select the netX Driver node in Driver folder in the Navigation Area, and select the port for the USB connection to the COMX module. Click Save and Apply (just click OK if Apply is grayed out):

IO Device: COMX	DMX_100XX_RE_EIS[COMX 100XX-RE/EIS]<192.168.10.1> 100XX-RE/EIS er GmbH	Device ID: Vendor ID:	0x0103 0x011B	
Navigation Area Settings Settings Priver Pevice Assignment Firmware Download Configuration General Electronic Keying Connection Assembly Device Settings Description Device Info	netX Driver USB/RS232 Connection TCP Connection I Image: Colspan="2">Enable USB/RS232 Connector (Restart of ODM required) Select Port: COM10 Port Configuratio COM10 Image: Disable Port COM9 Baud Rate: 115.2 kBit/s Byte Size: 8 Byte Stop Bits: 1 Stopbit Parity: No Parity Send Timeout: 1000 ms Keep Alive Timeout: 2000 Reset Timeout: 10000 ms Keep Alive Timeout: 2000	v v v ms		
치▷ Disconnected ① Data Set		Restore	Cancel App	Save All

Note: You can Check Windows Device Manager in order to identify which COM port provides the connection to the Hilscher COMX module:



Click the Device Assignment under Driver folder in the Navigation Area. Assign the netX Driver to the detected COMX module by checking the checkmark box next to the detected device, and click Apply:

	IX 100XX-RE/EIS her GmbH					Device ID: Vendor ID:	0x0103 0x011B	
lavigation Area 📃			1	Device Ass	ignment			
Settings 🔄 Driver	Scan progress: 3/3 D	evices (Current device: -)						
netX Driver Device Assignment	Device selection:	suitable only					L	Sca
Firmware Download	Device	Hardware Port 0/1/2/3	Slot nu	Serial num	Driver	Channel Protocol	Access path	
 Configuration General Electronic Keying Connection Assembly Device Settings Description Device Info 		RE Ethernet/Ethernet/-/-	n/a	20311	netX Driver	EtherNet/IP Adapter	\coms_afx0_ch0	
	Access path:	854C8CC7+F333-4135-8	405-6E12FC88	BEE62}\COM8_	_cifX0_Ch0	OK Can	cel Apply	Help

When used with Turbo PMAC, the reset line is released too fast for some Hilscher COMX modules, which puts them in a boot mode. This can prevent the device from being detected by Sycon.NET software. Make sure the device receives a system wide reset using the PMAC suggested M-variables ulSystemCommandCOS and HSF_RESET registers as shown here.

Note

 $SCtrl_ulSystemCommandCOS = \$55AA55AA$

HCSC_HSF_RESET=1

Note that ACC-72EX Setup Assistant software automatically resets the cards if it cannot detect the identification cookie.

retDevice - Configuration CO	OMX_RE_EIS[COMX RE/E	EIS]<192.168.1.210>				
	(RE/EIS er GmbH			Device ID: Vendor ID:	0x0103 0x011B	FDT
Navigation Area 📃			General			
Settings	Description:	OMX_RE_EIS				
netX Driver Device Assignment Firmware Download Configuration Connection Assembly Device Settings Consectings Description Device Info	IP Settings DHCP BootP Fixed Addresses IP Address: Network mask: Gateway: Note: The priority set Operation mode:	192 168 1 210 255 255 255 0 0 0 0 0 0 quence is DHCP, BootP, Fixed. All capable, Auto Negotiation et al. 1	enabled			T
				ОК	Cancel Appl	Help
Disconnected 🚺 Data Set						

Set the IP address for the COMX module in the General Configuration window:

Set Connections:

Navigation Area Connection Settings Connection name: Connection Maxigation Area Connection name: Connection netX Driver Originator to Target Originator to Target Device Assignment Firmware Download RT transfer format: 32-bit run/idle header Configuration General Target to Originator RT transfer format: 32-bit run/idle header Device Settings Note: The max. process IO data length depends on existance of run/idle header (02T, T2O) Note: The max. process IO data length depends on existance of run/idle header (02T, T2O) OK Cancel Apply Help		OMX_RE_EIS[COMX RE/EI K RE/EIS Ier GmbH	5]<192.168.1.210>	Device ID: Vendor ID:	0x0103 0x011B	
Inver Connection name: Connection netX Driver Originator to Target Device Assignment Originator to Target Firmware Download RT transfer format: 32-bit run/idle header Image: Connection RT transfer format: 32-bit run/idle header Image: Connection Target to Originator General Target to Originator Electronic Keying RT transfer format: 32-bit run/idle header Optice Settings RT transfer format: 32-bit run/idle header Device Settings Note: The max. process IO data length depends on existance of run/idle header (02T, T2O) Device Info Device Info				ection		
Configuration RT transfer format: 32-bit run/idle header General Electronic Keying Connection Arget to Originator Assembly Device Settings Description Note: The max. process IO data length depends on existance of run/idle header (02T, T2O) Device Info Device Info	Driver netX Driver		Connect1			
Electronic Keying Connection Assembly Device Settings Description Device Info	Configuration		32-bit run/idle header	•		
Description Device Info	Connection		32-bit run/idle header	•		
	Description	Note: The max. process I	D data length depends on existance of run/id	lle header <mark>(</mark> 02T, T2O)		
				ОК	Cancel App	ly Help

Set Instance IDs and Data lengths in the Assembly window. 240 is the maximum length for the CompactLogix 1769-L18ERM-BB1B controller.

🕨 netDevice - Configuration	n COMX_100XX_RE_	EIS[COMX 10	00XX-RE/EIS]<192.168.1.210>	_		_	- 0 X
	OMX 100XX-RE/EIS ilscher GmbH				Device ID: Vendor ID:	0x0103 0x011B	FDI
Navigation Area				Assembly			
 Settings Driver netX Driver Device Assignment Firmware Download Configuration General Electronic Keying Connection Assembly Device Settings Description Device Info 		ices: Connect1 Connect1	Connection name	Instance ID 101 100 2	Data length 240	Min. length C	
					ОК	Cancel Ap	pply Help
Disconnected 🚺 Data S	Set						

RSLogix 5000 Setup

RSLogix 5000 version 20 is used in this example.

Launch RSLogix, and click on <u>Who</u> Active in the Communications pull down menu to find the CompactLogix controller:

👸 R:	🔀 RSLogix 5000								
File	Edit	View	Search	Logic	Communications Tools				
					Who Active				
			Select Recent Path						
No Controller 🛛 🗸 🗖 RUN		UN		<u>G</u> o Online					
No Forces				<u>U</u> pload	ł				

👹 Who Active	
✔ Autobrowse Refresh ✔ Autobrowse Refresh ✔ Workstation, LT-TECHSUPW703 Intra Gateways, Ethernet ✔ Station, LT-TECHSUPW703 Intra Gateways, Ethernet ✔ Station, LT-TECHSUPW703 Intra Gateways, Ethernet Ⅰ I 192.168.0.20, 1769-L18ERM-BB1B CompactLogix Processor, 1769-L18ERM/A LOGIX5318ERM Ⅰ 1 192.168.1.200, 1769-L18ERM-BB1B CompactLogix Processor, 1769-L18ERM/A LOGIX5318ERM Ⅰ 1 192.168.1.200, 1769-L18ERM-BB1B CompactLogix Processor, 1769-L18ERM/A LOGIX5318ERM Ⅰ 1 192.168.1.210, COMX RE/EIM, COMX 100XX-RE/EIM Ⅰ 1 192.168.1.210, COMX RE/EIM, COMX 100XX-RE/EIM Ⅰ 192.168.1.210, COMX RE/EIM, COMX 100XX-RE/EIM	Go Online Upload Download Update Firmware Close Help
Path: AB_ETHIP-1\192.168.1.200 Path in Project: <none></none>	Set Project Path Clear Project Path

Select the controller, and click the Go Online button to test communication:

The Controller OK indicator box should change to green like below:

ß	RSLog	ix 5000) - Ethe	rNetIPTe	st [1769-	L18E
Ħ	File	Edit	View	Search	Logic	Con
						Ī
Pro	ogram		04	Progra	m Mode	
No	Forces		▶	Contro		
No	Edits		2	Battery		
				= 1/0 NG	ot Hespor	haing

Next, install the EDS file for the Hilscher COMX slave of the ACC-72EX. Go to the Tools pull down menu, and select EDS Hardware Installation Tool:

D 21	0.12	
s (Тос	ls Window Help
4		Options
		Security
	Ģ	Documentation <u>L</u> anguages
1		Import •
		Export •
	9	EDS Hardware Installation Tool
		Motion
		Custom Tools
	đ	Co <u>n</u> trolFLASH

Click Next:



Select the Register an EDS file(s) radial button:

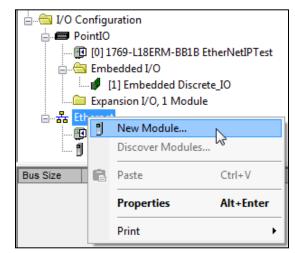


Browse to and select the Hilsher EDS file. EDS files can be downloaded at hilscher.com at http://www.hilscher.com/hcuk/support_software.html . Click Next.

Registration Electronic Data Sheet file(s) will b Automation applications.	be added to your system for use in Rockwell
 Register a single file 	
C Register a directory of EDS files	Look in subfolders
Named:	
C:\Users\techsupport\HILSCHER C	COMX-RE EIS V1.1.EDS Browse
If there is an icon file (ico) then this image will be assoc	with the same name as the file(s) you are registering stated with the device.

Follow the directions in the remaining windows for finishing the EDS installation.

The next step will apply the EDS installation, but first the controller needs to be offline. Click the <u>Go</u> Offline selection under the Communications tab (<u>Go</u> Offline is displayed when the controller is online, and <u>Go</u> Online is displayed when offline). Under I/O Configuration in the Controller Organizer, right-click on Ethernet, and select New Module...:



Scroll down to and select the COMX slave module, and press Create:

Select Mo	odule Type 19 Module Discovery Favori	ies			
E	inter Search Text for Module Tj	Clear Filters			Show Filters 🛛
	Catalog Number	Description	Vendor	Category	*
	1336T-FORCEDriveStd-E 1397DigitalDCDrive-EN1 150 SMC Rex-E 150-SMCCialogPlus-EN1 153x-10x 153x.10x 1557 1715-AENTR 1715-AENTR 1732E-12X4M12QCDR	AC Drive, PLC Comm Adapter via 1203-EN1 AC Drive, Standard Adapter via 1203-EN1 DC Drive via 1203-EN1 Smart Motor Controller via 20-COMM-E Smart Motor Controller via 1203-EN1 COMX RE/EIS COMX RE/EIS COMX RE/EIM 1557 Medium Voltage AC Drive 1715 Ethemet Adapter, Twisted Pair Media 12 Point Input/4 Point Output 24V DC Quick Connect	Allen-Bradley Allen-Bradley Allen-Bradley Allen-Bradley Allen-Bradley Hilscher GmbH Hilscher GmbH Allen-Bradley Allen-Bradley	Drive Drive Drive Drive Drive Communications Adapter Communications Adapter SCANport Drives on EtherNet/IP Communication Digital	
23	1732E-16CFGM12 1732E-16CFGM12QCR 1732E-16CFGM12QCWR 1732E-16CFGM12R 1732E-16CFGM12W 1732F-16CFGM12W	EtherNet/IP 16 Point Self-configuring 24VDC 16 Point 24V DC Self-Configuring Quick Connect, 2-Port 16 Point 24V DC Self-Configuring Quick Connect Weld 16 Point 24V DC Self-Configuring, 2-Port EtherNet/IP WeldBlock 16 Point Self-configuring 24VDC	Allen-Bradley Allen-Bradley Allen-Bradley Allen-Bradley Allen-Bradley	Digital Digital Digital Digital Dinital	Add to Favorites
	Close on Create			Creat	e Close Help

The created entry should appear under Ethernet in the Control Organizer. Right-click on it, and select Properties:

153x.10x COMX_RE_EIS						
		Print	•			
		Properties	Alt+Enter			
Embedded I/		Cross Reference	Ctrl+E			
😰 [0] 1769-L18		Delete	Del			
PointIO	B	Paste	Ctrl+V			
Trends	Ē,	Сору	Ctrl+C			
💮 🗰 Module-Defined	Ж	Cut	Ctrl+X			
🕀 🙀 Predefined		Discover Modules		≡		
Image Strings Image Add-On-Defined	đ	New Module				

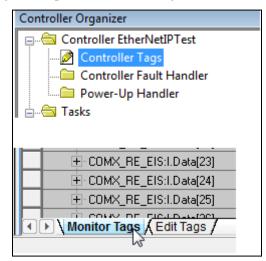
Under the General tab, set the IP address of the ACC-72EX:

I Module Proper	ties: Local (153x.10x 1.1)	
General Conne	ction Module Info Internet Protocol Port Configuration	Network
Туре:	153x.10x COMX RE/EIS	
Vendor:	Hilscher GmbH	
Parent:	Local	
Name:	COMX_RE_EIS	Ethernet Address
Description:		Private Network: 192.168.1. 210 -
		O IP Address:
		─ Host Name:
	-	
- Module Defini	lion	
Revision:	1.1	
Electronic Ke	ying: Exact Match	
Connections:	Exclusive Owner , Input_CP=101, Output	
	, , _ , , , _	
	Change	
	_	
Status: Offline		OK Cancel Apply Help

Check the settings under the Connection tab:

Module Properties: Local (153x.10x 1.1)						
General Connection Module Info Internet Protocol Por	t Configuration Network					
Name	Requested Packet Interval (RPI) (ms)	Input Type	Input Trigger			
Exclusive Owner , Input CP = 101, Output CP = 100	20.0 🜩 1.0 - 3200.0	Unicast 🗨	Cyclic			
 Inhibit Module Major Fault On Controller If Connection Fails While in Rui Module Fault 	n Mode					
Status: Offline	ок	Cancel	Apply Help			

Double-click on Controller Tags, and open the Monitor Tags tab:



Click on "+" to expand the input data entries. Now input values from the ACC-72EX can be seen in the Value column (controller must be online).

		Favorites Add-On A	Alarms 🔏 Bit 🔏 Timer/Counter 🔏 Inp	ut/Output 🚶 Compa	re 🔏 Compute/Mat	h 🕻 Move/Logical /
Controller Organizer 🛛 👻 🕂 🗙	s	cope: 🔞EtherNetIPTest 👻 Show: A	ll Tags			
Controller EtherNetIPTest		Name == △	Value 🗧	Force Mask 🔹 🗲	Style	Data Type
Controller Tags		-COMX_RE_EIS:I	{}	{}		_011B:153x10x
Power-Up Handler		-COMX_RE_EIS:I.ConnectionFault	0		Decimal	BOOL
		-COMX_RE_EIS:I.RunMode	1		Decimal	BOOL
🚊 🧔 MainTask		- COMX_RE_EIS:I.Data	{}	{}	Decimal	INT[240]
📄 🚭 MainProgram		COMX_RE_EIS:I.Data[0] €	24		Decimal	INT
Program Tags			24		Decimal	INT
MainRoutine			24		Decimal	INT
Unscheduled Programs			24		Decimal	INT
🖮 🔄 Motion Groups		COMX_RE_EIS:I.Data[4] E	24		Decimal	INT
Ungrouped Axes		COMX_RE_EIS:I.Data[5] €	24		Decimal	INT
Add-On Instructions		← COMX_RE_EIS:I.Data[6]	24		Decimal	INT
🖮 🔄 Data Types		COMX_RE_EIS:I.Data[7] €	24		Decimal	INT
🕀 🛄 User-Defined		COMX_RE_EIS:I.Data[8] E	24		Decimal	INT
🕀 🛄 Strings		COMX_RE_EIS:I.Data[9] €	24		Decimal	INT
Add-On-Defined			24		Decimal	INT
🗄 🙀 Predefined		COMX_RE_EIS:I.Data[11]	24		Decimal	INT
Module-Defined		COMX_RE_EIS:I.Data[12]	24		Decimal	INT
I/O Configuration		COMX_RE_EIS:I.Data[13]	24		Decimal	INT
PointIO		COMX_RE_EIS:I.Data[14]	24		Decimal	INT
[] [0] 1769-L18ERM-BB1B		COMX_RE_EIS:I.Data[15] E	24		Decimal	INT
Embedded I/O		COMX_RE_EIS:I.Data[16]	24		Decimal	INT
		COMX_RE_EIS:I.Data[17]	24		Decimal	INT
			24		Decimal	INT

Click on "+" to expand the output data entries. The values seen in the Value column should now be seen as inputs in the ACC-72EX. Values can be changed here manually, or in program logic such as in the ladder logic example that follows.

Name 📃 🗅	Value 🗲	Force Mask 🛛 🗲	Style	Data Type
COMX_RE_EIS:I.Data[238] E	23		Decimal	INT
E - COMX_RE_EIS:I.Data[239]	23		Decimal	INT
- COMX_RE_EIS:0	{}	{}		_011B:153x10x
-COMX_RE_EIS:0.Data	{}	{}	Decimal	INT[240]
	24		Decimal	INT
E COMX_RE_EIS:0.Data[1]	24		Decimal	INT
COMX_RE_EIS:0.Data[2] E	24		Decimal	INT
E - COMX_RE_EIS:0.Data[3]	24		Decimal	INT
COMX_RE_EIS:0.Data[4] E	24		Decimal	INT
COMX_RE_EIS:0.Data[5] E	24		Decimal	INT
∃-COMX_RE_EIS:0.Data[6]	24		Decimal	INT
E COMX_RE_EIS:0.Data[7]	24		Decimal	INT
E COMX_RE_EIS:0.Data[8]	24		Decimal	INT
COMX_RE_EIS:0.Data[9] E	24		Decimal	INT
E COMX_RE_EIS:0.Data[10]	24		Decimal	INT
COMX_RE_EIS:0.Data[11]	24		Decimal	INT
E COMX_RE_EIS:0.Data[12]	24		Decimal	INT
E COMX_RE_EIS:0.Data[13]	24		Decimal	INT
COMX_RE_EIS:0.Data[14]	24		Decimal	INT
E COMX_RE_EIS:0.Data[15]	24		Decimal	INT
COMX_RE_EIS:0.Data[16] E	24		Decimal	INT
E COMX_RE_EIS:0.Data[17]	24		Decimal	INT
COMX_RE_EIS:0.Data[18]	24		Decimal	INT

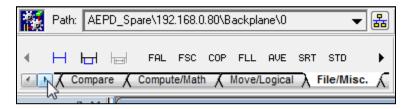
The following is an example which uses a "Copy" function to transfer all of the input values into corresponding output values. Double-click on MainRoutine:

Offline □		_Spare\192.168.0.80\Backplane\0	
Controller Organizer Controller AEPD_Spare Controller Tags Controller Fault Handler Power-Up Handler Tasks MainTask MainProgram Program Tags MainRoutine Unscheduled Programs	- # X	MainProgram - MainRoutine* Image: Constraint of the state	

Click to select the top rung:

📕 MainProgram - MainRou	ine*	
电 55 56 00 00	i db ab 💌 <ab></ab>	
e e (End)		

Click on the right arrow as needed to bring into view the File/Misc. ladder entries tab:



Click on the File/Misc. tab, and then drag and drop COP (copy) onto a rung. Look for a green dot to appear on the left side of the rung when the cursor is hovered there, and then drop the COP function. To copy all the ACC-72EX inputs into corresponding CopactLogix outputs, set Source to COMX_RE_EIS:I.Data[0], Dest to COMX_RE_EIS:O.Data[0], and Length to 240:

Controller Organizer - 🗸 🗸 🗙		
Controller EtherNetiP lest		
- 🖉 Controller Tags		COP
Controller Fault Handler	0	Copy File
Power-Up Handler		Source COMX_RE_EIS:LData[0] Dest COMX_RE_EIS:0.Data[0]
🚊 🔄 Tasks		Length 240
🚊 🧔 MainTask		
🚊 🚔 MainProgram		
🖉 Program Tags		
MainRoutine	(End)	
Unscheduled Programs		

COMX Test PLC

The following Turbo PMAC PLCs can be used to test the communication of the COMX module. Run PLC 1 and check the variable M_CommErrorFlag. If it is =0 after PLC 1 finishes, the COMX module is communicating properly.

```
CLOSE
END GAT
DEL GAT
#include "M-VariableDefinition $6C000.pmc"
#define M CommErrorFlag P1
#define timer i6612
#define msec *8388608/i10while(i6612>0)endwhile
OPEN PLC 1 CLEAR
DISABLE PLC 2..31 // Disable all other tasks
M SCtrl ulSystemCommandCOS=$55AA55AA // Reset token for MASTER Unit
M HCSC HSF RESET=1 // Reset bit, token required for reset to complete
M CommErrorFlag=0
timer = 4000 msec // Reset Time-out Timer
WHILE (M CommErrorFlag=0 AND M HCSC NSF READY=0) // Wait for reset to complete
IF (timer<0) // Check for reset timeout
M CommErrorFlag = 1
ENDIF
ENDWHILE
IF (M CommErrorFlag=0) //
WHILE (M CCO RCX COMM COS RUN=0) // wait for comm tasks to
// start on COMX modules
M HCCC0_HCF_NETX_COS_ACK = M_HCCC0_HCF_NETX_COS_ACK ^ 1
// Toggle Communication Channel 0's Change of State Acknowledge bit in
\prime\prime order to read the CCO RCX COMM COS RUN which is a part of Communication
// Channel 0 State Register
enable plc 2
ENDWHILE
ENDIF
DISABLE PLC 1
CLOSE
open plc2 clr
timer =4000 msec // Reset Time-out Timer
//M CCO RCX APP COS APP READY=1
       CSC_HSF_HOST_COS_CMD = M_HCSC_NSF_HOST_COS_ACK)
M_CC0_RCX_APP_COS_BUS_ON=1 // Setting the Bus On flag for 1st ACC-72EX
IF (M HCSC HSF HOST COS CMD
       M CCO RCX APP COS BUS ON ENABLE=1
 M_HCSC_HSF_HOST_COS_CMD
                               = M HCSC HSF HOST COS CMD^1
ENDIF
timer = 1000 msec // Reset Time-out Timer
ENABLE PLC 28
ENABLE PLC 10
ENABLE PLC 11
ENABLE PLC 26
disable plc2
close
OPEN PLC 28 CLEAR
M CCO ulDeviceWatchdog = M CCO ulHostWatchdog // copies the host watchdog content
CLOSE
OPEN PLC 10 CLEAR
IF (M HCCC0 HCF PD0 OUT CMD = M HCCC0 NCF PD0 OUT ACK) // Making sure the ACK flag matches the
CMD
```

M_HCCC0_HCF_PD0_OUT_CMD ENDIF	= M_HCCC0_HCF_PD0_OUT_CMD^1 // Toggling the CMD flag (^: XOR)
CLOSE	
	<pre>0 = M_HCCC0_HCF_PD0_IN_ACK) // If CMD flag and ACK flags are _ACK = M_HCCC0_HCF_PD0_IN_ACK ^ 1 // toggle the acknowledge bit</pre>
CLOSE	
i5=2 Ena plc1	
	;Read Address ;address of M90 ;address of m91

The above PLCs require the following header files:

<pre>// MacroNameDefinition_\$6C000.h</pre>			
#define M_SI_abCookie_0_	M5000		
#define M_SI_abCookie_1_	M5001		
#define M_SI_abCookie_2_	M5002		
#define M_SI_abCookie_3_	M5003		
#define M_SI_ulDpmTotalSize	M5004		
#define M_SI_ulDeviceNumber	M5005		
#define M_SI_ulSerialNumber	M5006		
#define M_SI_ausHwOptions_0_	M5007		
#define M_SI_ausHwOptions_1_	M5008		
#define M_SI_ausHwOptions_2_	M5009		
#define M_SI_ausHwOptions_3	M5010		
#define M_SI_usManufacturer	M5011		
#define M_SI_usProductionDate	M5012		
#define M_SI_ulLicenseFlags1	M5013		
#define M SI ulLicenseFlags2	M5014		
#define M_SI_usNetxLicenseID	M5015		
#define M SI usNetxLicenseFlags		M5016	
#define M SI usDeviceClass	M5017		
#define M_SI_bHwRevision	M5018		
#define M_SI_bHwCompatibility	M5019		
#define M SI bDevIdNumber	M5020		
#define M SCI bChannelType	M5021		
#define M SCI bSizePositionOfHandsha	ke		M5022
#define M SCI bNumberOfBlocks	M5023		
#define M_SCI_ulSizeOfChannel	M5024		
#define M_SCI_usSizeOfMailbox	M5025		
#define M SCI usMailboxStartOffset		M5026	
#define M HCI bChannelType	M5027		
#define M HCI ulSizeOfChannel	M5028		
#define M CC0I bChannelType	M5029		
#define M CC0I bChannelId	M5030		
#define M CCOI bSizePositionOfHandsh	ake		М5031
#define M CC0I bNumberOfBlocks		M5032	
#define M CC0I ulSizeOfChannel		M5033	
#define M CC0I usCommunicationClass		M5034	
#define M CCOI usProtocolClass		M5035	
#define M CCOI usConformanceClass		M5036	
	M5037		
#define M CC1I bChannelId	M5038		
#define M CC1I bSizePositionOfHandsh	ake		М5039
<pre>#define M_CC1I_bSizePositionOfHandsh</pre>	ake		M5039

#define M CC1I bNumberOfBlocks			
		M5040	
<pre>#define M_CC1I_ulSizeOfChannel</pre>		M5041	
#define M CC1I usCommunication	Class	M5042	
<pre>#define M_CC1I_usProtocolClass</pre>		M5043	
#define M CC1I usConformanceCl	ass	M5044	
#define M_CC2T_bChannelType	M5045		
	115045		
<pre>#define M_CC2I_bChannelType #define M_CC2I_bChannelId #define M_CC2I_bCianDefine</pre>	M5046		
#define M CC2I bSizePositionOf	Handshake		M5047
		MEOIO	
#define M_CC2I_bNumberOfBlocks		M5048	
#define M CC2I ulSizeOfChannel		M5049	
#define M CC2I usCommunication	Class	M5050	
<pre>#define M_CC2I_usProtocolClass</pre>		M5051	
#define M CC2I usConformanceCl	ass	M5052	
	MEGEO	110002	
#define M_CC31_bChannelType	M5053		
<pre>#define M_CC3I_bChannelType #define M_CC3I_bChannelId</pre>	M5054		
#define M_CC3I_bSizePositionOf	Handehako		M5055
#deline M_ccoi_boizerosicionor	lialiusliake		115055
#define M CC3I bNumberOfBlocks		M5056	
#define M CC3I ulSizeOfChannel		M5057	
#define M_CC3I_usCommunication		M5058	
#define M CC3I usProtocolClass		M5059	
#define M CC3I usConformanceCl		M5060	
#define M_AC0I_bChannelType	M5061		
#define M ACOI bChannelId	M5062		
#define M_ACOT_bonanneriu	113002		
#define M_ACOI_bSizePositionOf	Handshake		M5063
<pre>#define M_AC0I_bNumberOfBlocks #define M_AC0I_ulSizeOfChannel #define M_AC1I_bChannelType</pre>		M5064	
#define M_ACUI_ulSizeOfChannel		M5065	
<pre>#define M_AC11_bChannelType #define M_AC11_bChannelId</pre>	M5066		
	MEOCO		
#define M AC1I bSizePositionOf	Handshake		M5068
#define M AC11 bNumberOfBlocks		M5069	
#define M AC1I_ulSizeOfChannel		M5070	
#define M SCtrl ulSystemComman	dCOS	M5071	
<pre>#define M_SStat_ulSystemCOS</pre>	M5072		
#define M SStat ulSystemStatus		M5073	
#define M_SStat_ulSystemError	M5074		
#define M SStat ulBootError	M5075		
#define M SStat ulTimeSinceSta	rt	M5076	
		110070	
#define M_SStat_usCpuLoad	M5077		
#define M SStat ulHWFeatures	M5078		
#define M SSMB usPackagesAccep	tod	M5079	
		MJ079	
#define M_SSMB_ulDest	M5080		
#define M_SSMB_ulSrc	M5081		
#define M_SSMB_ulSrc	M5081		
#define M_SSMB_ulSrc #define M_SSMB_ulDestId	M5081 M5082		
#define M_SSMB_ulSrc #define M_SSMB_ulDestId			
#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId	M5082 M5083		
#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen	M5082 M5083 M5084		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId</pre>	M5082 M5083		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId</pre>	M5082 M5083 M5084 M5085		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState</pre>	M5082 M5083 M5084 M5085 M5086		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd</pre>	M5082 M5083 M5084 M5085		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd</pre>	M5082 M5083 M5084 M5085 M5086		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ulRout</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ulRout #define M_SSMB_ultData0</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ulRout #define M_SSMB_ultData0</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData4</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5094		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ulLoata0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData4</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5094 M5095		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData4</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5094 M5095 M5096		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5094 M5095		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData4</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5094 M5095 M5096 M5097		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8</pre>	M5082 M5083 M5084 M5085 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5093 M5094 M5095 M5096 M5097 M5098		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData8</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5093 M5094 M5095 M5096 M5097 M5098 M5099		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8</pre>	M5082 M5083 M5084 M5085 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5093 M5094 M5095 M5096 M5097 M5098		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData7 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData9 #define M_SSMB_ultData10</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5094 M5094 M5095 M5096 M5097 M5098 M5099 M5099 M5099 M5099		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulCmd #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData9 #define M_SSMB_ultData10 #define M_SSMB_ultData11</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5094 M5095 M5094 M5095 M5096 M5097 M5098 M5099 M5100 M5101		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData7 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData9 #define M_SSMB_ultData10</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5094 M5094 M5095 M5096 M5097 M5098 M5099 M5099 M5099 M5099		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulState #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData9 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData11 #define M_SSMB_ultData12</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5095 M5096 M5097 M5098 M5098 M5099 M5100 M5101 M5102		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulCmd #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData7 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData10 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData13</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5095 M5096 M5097 M5098 M5099 M5100 M5101 M5102 M5103		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulCmd #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData1 #define M_SSMB_ultData1 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData14</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5095 M5096 M5097 M5098 M5099 M5100 M5101 M5102 M5103 M5104		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulCmd #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData7 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData8 #define M_SSMB_ultData10 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData13</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5095 M5096 M5097 M5098 M5099 M5100 M5101 M5102 M5103		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulId #define M_SSMB_ulId #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData1 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData14 #define M_SSMB_ultData14 #define M_SSMB_ultData14 #define M_SSMB_ultData15</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5095 M5096 M5097 M5098 M5099 M5100 M5101 M5101 M5102 M5103 M5104 M5105		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData10 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData14 #define M_SSMB_ultData15 #define M_SSMB_ultData15 #define M_SSMB_ultData16</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5093 M5093 M5094 M5095 M5095 M5096 M5097 M5098 M5099 M5100 M5101 M5102 M5103 M5104 M5105 M5106		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulState #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData10 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData14 #define M_SSMB_ultData15 #define M_SSMB_ultData14 #define M_SSMB_ultData15 #define M_SSMB_ultData16 #define M_SSMB_ultData17</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5095 M5096 M5097 M5098 M5099 M5100 M5101 M5101 M5102 M5103 M5104 M5105		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulState #define M_SSMB_ulExt #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData10 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData14 #define M_SSMB_ultData15 #define M_SSMB_ultData14 #define M_SSMB_ultData15 #define M_SSMB_ultData16 #define M_SSMB_ultData17</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5089 M5090 M5091 M5092 M5093 M5093 M5093 M5094 M5095 M5095 M5096 M5097 M5098 M5099 M5100 M5101 M5102 M5103 M5104 M5105 M5106		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ulCmd #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData10 #define M_SSMB_ultData10 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData14 #define M_SSMB_ultData15 #define M_SSMB_ultData15 #define M_SSMB_ultData16 #define M_SSMB_ultData17 #define M_SSMB_ultData17 #define M_SSMB_ultData17</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5093 M5094 M5095 M5096 M5097 M5098 M5099 M5100 M5101 M5102 M5103 M5104 M5105 M5106 M5107 M5108		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData10 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData14 #define M_SSMB_ultData15 #define M_SSMB_ultData15 #define M_SSMB_ultData16 #define M_SSMB_ultData17 #define M_SSMB_ultData18 #define M_SSMB_ultData18 #define M_SSMB_ultData18 #define M_SSMB_ultData18</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5093 M5094 M5095 M5095 M5095 M5097 M5098 M5099 M5100 M5101 M5102 M5103 M5104 M5105 M5105 M5106 M5107 M5108 M5109		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulSrcId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulCnd #define M_SSMB_ulCnd #define M_SSMB_ulCnt #define M_SSMB_ulCnta1 #define M_SSMB_ultData2 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData10 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData14 #define M_SSMB_ultData15 #define M_SSMB_ultData15 #define M_SSMB_ultData17 #define M_SSMB_ultData17 #define M_SSMB_ultData17 #define M_SSMB_ultData17 #define M_SSMB_ultData18</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5093 M5094 M5095 M5096 M5097 M5098 M5099 M5100 M5101 M5102 M5103 M5104 M5105 M5106 M5107 M5108		
<pre>#define M_SSMB_ulSrc #define M_SSMB_ulDestId #define M_SSMB_ulDestId #define M_SSMB_ulLen #define M_SSMB_ulLen #define M_SSMB_ulState #define M_SSMB_ulState #define M_SSMB_ulCmd #define M_SSMB_ultData0 #define M_SSMB_ultData1 #define M_SSMB_ultData1 #define M_SSMB_ultData3 #define M_SSMB_ultData3 #define M_SSMB_ultData4 #define M_SSMB_ultData5 #define M_SSMB_ultData5 #define M_SSMB_ultData6 #define M_SSMB_ultData7 #define M_SSMB_ultData7 #define M_SSMB_ultData8 #define M_SSMB_ultData10 #define M_SSMB_ultData11 #define M_SSMB_ultData12 #define M_SSMB_ultData12 #define M_SSMB_ultData13 #define M_SSMB_ultData13 #define M_SSMB_ultData14 #define M_SSMB_ultData15 #define M_SSMB_ultData15 #define M_SSMB_ultData16 #define M_SSMB_ultData17 #define M_SSMB_ultData18 #define M_SSMB_ultData18 #define M_SSMB_ultData18 #define M_SSMB_ultData18 #define M_SSMB_ultData18</pre>	M5082 M5083 M5084 M5085 M5086 M5087 M5088 M5099 M5091 M5092 M5093 M5094 M5093 M5094 M5095 M5095 M5095 M5097 M5098 M5099 M5100 M5101 M5102 M5103 M5104 M5105 M5105 M5106 M5107 M5108 M5109		

<pre>#define M_SRMB_usWaitingPackages</pre>		M5111
#define M SRMB ulDest M5112		
#define M_SRMB_ulSrc M5113		
#define M SRMB ulDestId	M5114	
#define M_SRMB_ulSrcId M5115		
#define M SRMB ulLen M5116		
#define M_SRMB_ulId M5117		
#define M SRMB ulState M5118		
#define M SRMB_ulCmd M5119		
#define M_SRMB_ulExt M5120		
#define M_SRMB_ulRout M5121		
<pre>#define M_SRMB_ultData0 #define M_SRMB_ultData1</pre>	M5122	
	M5123	
#define M_SRMB_ultData2	M5124	
#define M_SRMB_ultData3	M5125	
#define M_SRMB_ultData4	M5126	
#define M_SRMB_ultData5	M5127	
#define M SRMB ultData6	M5128	
#define M_SRMB_ultData7	M5129	
#define M SRMB ultData8	M5130	
#define M_SRMB_ultData9	M5131	
#define M SRMB ultData10	M5132	
#define M SRMB ultData11	M5133	
#define M_SRMB_ultData12	M5134	
#define M SRMB_ultData12 #define M SRMB ultData13	M5134 M5135	
#define M SRMB_ultData13 #define M SRMB ultData14	M5135 M5136	
#define M_SRMB_ultData15	M5137	
#define M_SRMB_ultData16	M5138	
<pre>#define M_SRMB_ultData17</pre>	M5139	
#define M_SRMB_ultData18	M5140	
#define M_SRMB_ultData19	M5141	
#define M_SRMB_ultData20	M5142	
#define M HCSC bNetxFlags	M5143	
#define M HCSC NSF READY	M5144	
#define M_HCSC_NSF_ERROR	M5145	
#define M_HCSC_NSF_HOST_COS_ACK		M5146
#define M HCSC NSF NETX COS CMD		M5147
#define M HCSC NSF SEND MBX ACK		M5148
#define M_HCSC_NSF_SEND_MBX_ACK #define M_HCSC_NSF_RECV_MBX_CMD		M5149
	NE150	MJ149
#define M_HCSC_bHostFlags	M5150	
	M5151	
#define M_HCSC_HSF_BOOTSTART	M5152	
<pre>#define M_HCSC_HSF_HOST_COS_CMD</pre>		M5153
<pre>#define M_HCSC_HSF_NETX_COS_ACK</pre>		M5154
#define M_HCSC_HSF_SEND_MBX_CMD		M5155
#define M_HCSC_HSF_RECV_MBX_ACK		M5156
#define M HCCCO usNetxFlags	M5157	
#define M HCCC0 NCF COMMUNICATING		M5158
#define M HCCC0 NCF ERROR	M5159	
#define M HCCC0 NCF HOST COS ACK		M5160
#define M HCCC0 NCF NETX COS CMD		M5160
#define M HCCC0 NCF SEND MBX ACK		M5161 M5162
#define M HCCC0 NCF RECV MBX CMD		M5162 M5163
#define M_HCCC0_NCF_PD0_OUT_ACK		M5164
#define M_HCCC0_NCF_PD0_IN_CMD		M5165
#define M_HCCC0_NCF_PD1_OUT_ACK		M5166
<pre>#define M_HCCC0_NCF_PD1_IN_CMD</pre>		M5167
<pre>#define M_HCCC0_usHostFlags</pre>	M5168	
<pre>#define M_HCCC0_HCF_HOST_COS_CMD</pre>		M5169
#define M_HCCC0_HCF_NETX_COS_ACK		M5170
#define M HCCCO HCF SEND MBX CMD		M5171
#define M HCCC0 HCF RECV MBX ACK		M5172
#define M HCCCO HCF PD0 OUT CMD		M5173
#define M HCCCO HCF PDO IN ACK		M5174
#define M HCCCO HCF PD1 OUT CMD		M5175
#define M HCCCO HCF PD1 IN ACK		M5176
	M5177	1101.10
#define M_HCCC1_usNetxFlags	M5177	ME170
#define M_HCCC1_NCF_COMMUNICATING	NE1 20	M5178
#define M_HCCC1_NCF_ERROR	M5179	
<pre>#define M_HCCC1_NCF_HOST_COS_ACK</pre>		M5180
#define M_HCCC1_NCF_NETX_COS_CMD		M5181

#define M ACCIL NCP SIND MAX ACK M5182 #define M ACCIL NCP SIND MAX CMD M5183 #define M ACCIL NCP FNID ONT ACK M5184 #define M ACCIL NCP FNID NR ACMD M5185 #define M ACCIL NCP FNID IN CMD M5185 #define M ACCIL ACCIL EXC FNID IN CMD M5185 #define M ACCIL ACCIL EXC FNID IN CMD M5185 #define M ACCIL FCP FNIT CAS CAS ACK M5190 #define M ACCIL FCP FNIT CAS CAS ACK M5191 #define M ACCIL FCP FNIT CAS CAS ACK M5191 #define M ACCIL FCP FNIT CAT CAS M5193 #define M ACCIL FCP FNIT CAT CAS M5193 #define M ACCIL FCP FNIT CAT CAS M5193 #define M ACCIL FCP FNIT CAT CAS ACK M5193 #define M ACCIL FCP FNIT CAT CAS ACK M5193 #define M ACCIL FCP FNIT CAS ACK M5200 #define M ACCIL FCP FNIT CAS ACK M5201 #define M ACCIL FCP FNIT CAS ACK M5202 #define M ACCIL FCP FNIT CAS ACK M5204			
Adefine M_HOCCL BOCF_PRO_DOT_ACK MS184 Adefine M_HOCCL BOCF_PRO_DOT ACK MS186 Adefine M_HOCCL BOCF_PRO_DOT ACK MS180 Adefine M_HOCCL BOCF_PRO_DOT ACK MS180 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS180 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS181 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS181 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS180 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS181 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS201 Adefine M_HOCCL BOCF_BOCF_PRO_DOT CON_ACK MS201 Adefine M_HOCCL BOCF_BOCF_BOCF_CON_ACK MS201 Adefine M_HOCCL BOCF_BOCF_BOCK_CON_ACK MS204 Adefine M_HOCCL BOCF_BOCF_BOCK	#define M HCCC1 NCF SEND MBX ACK		М5182
Adefine M_HOCCL BOCF_PRO_DOT_ACK MS184 Adefine M_HOCCL BOCF_PRO_DOT ACK MS186 Adefine M_HOCCL BOCF_PRO_DOT ACK MS180 Adefine M_HOCCL BOCF_PRO_DOT ACK MS180 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS180 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS181 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS181 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS180 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS181 Adefine M_HOCCL BOCF_PRO_DOT CON_ACK MS201 Adefine M_HOCCL BOCF_BOCF_PRO_DOT CON_ACK MS201 Adefine M_HOCCL BOCF_BOCF_BOCF_CON_ACK MS201 Adefine M_HOCCL BOCF_BOCF_BOCK_CON_ACK MS204 Adefine M_HOCCL BOCF_BOCF_BOCK	#define M HCCC1 NCF RECV MBX CMD		M5183
Adefine M. HOCCI, NOT PRIO TO TACK M5185 Adefine M. ACCI, NOT PRIO TO TACK M5186 Adefine M. ACCI, AND PRIO TO TACK M5187 Adefine M. ACCI, AND PRIO TO TACK M5180 Adefine M. ACCI, AND PRIO TO TACK M5190 Adefine M. ACCI, AND PRIO TO TACK M5191 Adefine M. ACCI, AND PRIO TO TACK M5193 Adefine M. ACCI, AND PRIO TO TACK M5193 Adefine M. ACCI, AND PRIO TO TACK M5194 Adefine M. ACCI, AND PRIO TO AND M5195 M5197 Adefine M. ACCI, ACCI, AND MARKAK M5197 Adefine M. ACCC, ACC, NETR NO, ACK M5200 Adefine M. ACCC, ACC, NETR NO, ACK M5201 Adefine M. ACCC, ACC, NETR NO, ACK M5201 Adefine M. ACCC, ACC, PRO, NETX, ACK M5202 Adefine M. ACCC, ACC, PRO, NON M5203 Adefine M. ACCC, ACC, PRO, NON M5204 Adefine M. ACCC, ACC, PRO, NON M5204 Adefine M. ACCCC, BACC, PRO, PRO, ACK M5204	#define M HCCC1 NCF PD0 OUT ACK		м5184
Addrine M LUCCU TWC Prol Tot TACK M5186 Addrine M LUCCU TWC Prol TACK M5187 Addrine M LUCCU TWC Prol TACK M5188 Addrine M LUCCU TWC Prol TACK M5188 Addrine M LUCCU TWC Prol TACK M5189 Addrine M LUCCU TWC Prol TACK M5190 Addrine M LUCCU TWC Prol TACK M5191 Addrine M LUCCU TWC Prol TACK M5193 Addrine M LUCCU TWC Prol TA TACK M5194 Addrine M LUCCU TWC Prol TA TACK M5196 Addrine M LUCCU TWC Prol TA TACK M5196 Addrine M LUCCU TWC Prol TACK M5198 Addrine M LUCCU TWC Prol TACK M5201 Addrine M LUCCU TWC Prol TACK M5204	#define M HCCC1 NCE PD0 IN CMD		M5185
Adefine M. MCCCI NCC FDI IN CMD M5187 Adefine M. MCCCI LOC LUBOR COS ACC M5188 Adefine M. MCCCI NCC FWICK COS ACC M5183 Adefine M. MCCCI NCC FWICK COS ACC M5193 Adefine M. MCCCI NCC FWICK COS ACC M5193 Adefine M. MCCCI NCC FWICK COS ACC M5194 Adefine M. MCCCI NCC FWICK COS CMN M5196 Adefine M. MCCCI NCC FWICK COS ACC M5198 Adefine M. MCCCI NCC FWICK COS ACC M5198 Adefine M. MCCCI NCC FWICK COS ACC M5200 Adefine M. MCCCI NCC FWICK COS ACC M5200 Adefine M. MCCCI NCC FWICK COS ACC M5201 Adefine M. MCCCI NCC FWICK COS ACC M5203 Adefine M. MCCCI NCC FWICK COS ACC M5204 Adefine M. MCCCI NCC FWICK COS ACC M5204 Adefine M. MCCCI NCC FWICK COS ACC M5204 Adefine M. MCCCI NCCI SWICK COS ACC M5204 Adefine M. MCCCI NCCI SWICK COS ACC M5204 Adefine M. MCCCI NCCI SWICK COS ACC <	#define M_HCCC1_NCF_FD0_IN_CMD		
the fire M_HCCCC_ NET_GOS_COS_COS M5188 tefrim M_HCCCL_NET_NETX_COS_ACK M5190 tefrim M_HCCCL_NET_NETX_COS_ACK M5190 tefrim M_HCCCL_NET_NETX_COS_ACK M5191 tefrim M_HCCCL_NET_NETX_NEX_ACK M5193 tefrim M_HCCCL_NET_NEXT_NEXX M5193 tefrim M_HCCCL_NET_NEXT_NEXX M5193 tefrim M_HCCCL_NET_NEXT_NEXX M5193 tefrim M_HCCCL_NET_NEXX M5193 tefrim M_HCCCL_NET_NEXX M5194 tefrim M_HCCCL_NET_SANX M5194 tefrim M_HCCCL_NET_SANX M5194 tefrim M_HCCCL_NET_SANX M5194 tefrim M_HCCCL_NET_SANX M5194 tefrim M_HCCCL_NET_NEXX M5200 tefrim M_HCCCL_NET_NEXX M5201 tefrim M_HCCCL_NET_NEXX M5201 tefrim M_HCCCL_NET_NEXX M5203 tefrim M_HCCCL_NET_NEXX M5204 tefrim M_HCCCL_NET_NEXX M5204 tefrim M_HCCCL_NET_NEXX M5205 tefrim M_HCCCL_NET_NEXX M5206 tefrim M_HCCCL_NET_NEXX M5206 tefrim M_HCCCL_NET_NEXX M5204 <	#deline M_HCCCI_NCF_PDI_OUL_ACK		
#define M_ECCC1_HCG_NEX_COS_ACK M5100 #define M_ECCC1_HCG_NEX_MEX_CK M5130 #define M_ECCC1_HCG_NEX_CMC M5132 #define M_ECCC1_HCG_NEX_CMC M5134 #define M_ECCC1_HCG_NOT_CMD M5134 #define M_ECCC1_HCG_NOT_CMD M5134 #define M_ECCC1_HCG_NOT_CMD M5197 #define M_ECCC2_UNEX_LAISE M5197 #define M_ECCC2_NCF_NEXX_COS_CMD M5198 #define M_ECCC2_NCF_NEXX_COS_CMD M5198 #define M_ECCC2_NCF_NEXX_COS_CMD M5201 #define M_ECCC2_NCF_NEXX_COS_CMD M5203 #define M_ECCC2_NCF_FNEXX_COS_CMD M5203 #define M_ECCC2_NCF_FNEXX_COS_CMD M5204 #define M_ECCC2_NCF_FNEXX_COS_CMD M5205 #define M_ECCC2_NCF_FNEXX_COS_CMD M5204 #define M_ECCC2_NCF_FNE	#define M_HCCC1_NCF_PD1_IN_CMD		M5187
#define M_ECCC1_HCG_NEX_COS_ACK M5100 #define M_ECCC1_HCG_NEX_MEX_CK M5130 #define M_ECCC1_HCG_NEX_CMC M5132 #define M_ECCC1_HCG_NEX_CMC M5134 #define M_ECCC1_HCG_NOT_CMD M5134 #define M_ECCC1_HCG_NOT_CMD M5134 #define M_ECCC1_HCG_NOT_CMD M5197 #define M_ECCC2_UNEX_LAISE M5197 #define M_ECCC2_NCF_NEXX_COS_CMD M5198 #define M_ECCC2_NCF_NEXX_COS_CMD M5198 #define M_ECCC2_NCF_NEXX_COS_CMD M5201 #define M_ECCC2_NCF_NEXX_COS_CMD M5203 #define M_ECCC2_NCF_FNEXX_COS_CMD M5203 #define M_ECCC2_NCF_FNEXX_COS_CMD M5204 #define M_ECCC2_NCF_FNEXX_COS_CMD M5205 #define M_ECCC2_NCF_FNEXX_COS_CMD M5204 #define M_ECCC2_NCF_FNE	#define M_HCCC1_usHostFlags	M5188	
idefine M_HCCC1_HCC7_SEND_MEX_ACK M5191 idefine M_HCCC1_HC7_FD0_GTT_CND M5193 idefine M_HCCC1_HC7_FD0_GTT_CND M5193 idefine M_HCCC1_HC7_FD0_GTT_CND M5193 idefine M_HCCC1_HC7_FD0_GTT_CND M5194 idefine M_HCCC1_HC7_FD0_GTT_CND M5195 idefine M_HCCC1_HC7_FD0_GTT_CND M5197 idefine M_HCCC2_NCF_EXENCR M5197 idefine M_HCCC2_NCF_FD0_GTT_CNS M5197 idefine M_HCCC2_NCF_EXENCR M5199 idefine M_HCCC2_NCF_FD0_GTT_CNS M5200 idefine M_HCCC2_NCF_FD0_GTT_CNS M5201 idefine M_HCCC2_NCF_FD0_GTT_CNS M5204 idefine M_HCCC2_NCF_FD0_GTT_CNS M5204 idefine M_HCCC2_NCF_FD0_GTT_CNS M5204 idefine M_HCCC2_NCF_FD0_GTT_CNS M5206 idefine M_HCCC2_NCF_FD0_GTT_CNS M5206 idefine M_HCCC2_NCF_FD0_GTT_CNS M5206 idefine M_HCCC2_NCF_FD0_GTT_CNS M5210 idefine M_HCCC2_NCF_FD0_GTT_CNS M5211 idefine M_HCCC2_NCF_FD0_GTT_NCS M5213 idefine M_HCCC2_NCF_FD0_GTT_NCS M5216 idefine M_HCCC2_NCF_FD0_GTT_N	#define M_HCCC1_HCF_HOST_COS_CMD		M5189
idefine M_HCCC1_HCC7_SEND_MEX_ACK M5191 idefine M_HCCC1_HC7_FD0_GTT_CND M5193 idefine M_HCCC1_HC7_FD0_GTT_CND M5193 idefine M_HCCC1_HC7_FD0_GTT_CND M5193 idefine M_HCCC1_HC7_FD0_GTT_CND M5194 idefine M_HCCC1_HC7_FD0_GTT_CND M5195 idefine M_HCCC1_HC7_FD0_GTT_CND M5197 idefine M_HCCC2_NCF_EXENCR M5197 idefine M_HCCC2_NCF_FD0_GTT_CNS M5197 idefine M_HCCC2_NCF_EXENCR M5199 idefine M_HCCC2_NCF_FD0_GTT_CNS M5200 idefine M_HCCC2_NCF_FD0_GTT_CNS M5201 idefine M_HCCC2_NCF_FD0_GTT_CNS M5204 idefine M_HCCC2_NCF_FD0_GTT_CNS M5204 idefine M_HCCC2_NCF_FD0_GTT_CNS M5204 idefine M_HCCC2_NCF_FD0_GTT_CNS M5206 idefine M_HCCC2_NCF_FD0_GTT_CNS M5206 idefine M_HCCC2_NCF_FD0_GTT_CNS M5206 idefine M_HCCC2_NCF_FD0_GTT_CNS M5210 idefine M_HCCC2_NCF_FD0_GTT_CNS M5211 idefine M_HCCC2_NCF_FD0_GTT_NCS M5213 idefine M_HCCC2_NCF_FD0_GTT_NCS M5216 idefine M_HCCC2_NCF_FD0_GTT_N	#define M HCCC1 HCF NETX COS ACK		М5190
Hardine B, ROCCI RCF, POL IN, ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M5200 Haddine M, ROCCZ NCF, ENST, COS ACK M5201 Haddine M, ROCCZ NCF, ENST, COS ACK M5203 Haddine M, ROCCZ NCF, ENST, COS ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5203 Haddine M, ROCCZ NCF, PROV MAX ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5206 Haddine M, ROCCZ NCF, PROV MAX ACK M5207 Haddine M, ROCCZ HCF, BKNM, MAX ACK M5210 Haddine M, ROCCZ HCF, RSNM, MAX ACK M5211 Haddine M, ROCCZ HCF, PROV MAX ACK M5212 Haddine M, ROCCZ HCF, PROV MAX ACK M5213 Haddine M, ROCCZ HCF, PROV MAX ACK M5214 Haddine M, ROCCZ HCF, PROV MAX ACK M5215 Haddine M, ROCCZ HCF, PROV MAX ACK M5216 Haddine M, ROCCZ HCF, PROV MAX ACK M5217 Haddine M, ROCCZ HCF, PROV MAX ACK	#define M HCCC1 HCF SEND MBX CMD		M5191
Hardine B, ROCCI RCF, POL IN, ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M5200 Haddine M, ROCCZ NCF, ENST, COS ACK M5201 Haddine M, ROCCZ NCF, ENST, COS ACK M5203 Haddine M, ROCCZ NCF, ENST, COS ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5203 Haddine M, ROCCZ NCF, PROV MAX ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5206 Haddine M, ROCCZ NCF, PROV MAX ACK M5207 Haddine M, ROCCZ HCF, BKNM, MAX ACK M5210 Haddine M, ROCCZ HCF, RSNM, MAX ACK M5211 Haddine M, ROCCZ HCF, PROV MAX ACK M5212 Haddine M, ROCCZ HCF, PROV MAX ACK M5213 Haddine M, ROCCZ HCF, PROV MAX ACK M5214 Haddine M, ROCCZ HCF, PROV MAX ACK M5215 Haddine M, ROCCZ HCF, PROV MAX ACK M5216 Haddine M, ROCCZ HCF, PROV MAX ACK M5217 Haddine M, ROCCZ HCF, PROV MAX ACK	#define M HCCC1 HCF BECV MBX ACK		м5192
Hardine B, ROCCI RCF, POL IN, ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M5200 Haddine M, ROCCZ NCF, ENST, COS ACK M5201 Haddine M, ROCCZ NCF, ENST, COS ACK M5203 Haddine M, ROCCZ NCF, ENST, COS ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5203 Haddine M, ROCCZ NCF, PROV MAX ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5206 Haddine M, ROCCZ NCF, PROV MAX ACK M5207 Haddine M, ROCCZ HCF, BKNM, MAX ACK M5210 Haddine M, ROCCZ HCF, RSNM, MAX ACK M5211 Haddine M, ROCCZ HCF, PROV MAX ACK M5212 Haddine M, ROCCZ HCF, PROV MAX ACK M5213 Haddine M, ROCCZ HCF, PROV MAX ACK M5214 Haddine M, ROCCZ HCF, PROV MAX ACK M5215 Haddine M, ROCCZ HCF, PROV MAX ACK M5216 Haddine M, ROCCZ HCF, PROV MAX ACK M5217 Haddine M, ROCCZ HCF, PROV MAX ACK	#define M HCCC1 HCE PD0 OUT CMD		M5193
Hardine B, ROCCI RCF, POL IN, ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M5200 Haddine M, ROCCZ NCF, ENST, COS ACK M5201 Haddine M, ROCCZ NCF, ENST, COS ACK M5203 Haddine M, ROCCZ NCF, ENST, COS ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5203 Haddine M, ROCCZ NCF, PROV MAX ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5206 Haddine M, ROCCZ NCF, PROV MAX ACK M5207 Haddine M, ROCCZ HCF, BKNM, MAX ACK M5210 Haddine M, ROCCZ HCF, RSNM, MAX ACK M5211 Haddine M, ROCCZ HCF, PROV MAX ACK M5212 Haddine M, ROCCZ HCF, PROV MAX ACK M5213 Haddine M, ROCCZ HCF, PROV MAX ACK M5214 Haddine M, ROCCZ HCF, PROV MAX ACK M5215 Haddine M, ROCCZ HCF, PROV MAX ACK M5216 Haddine M, ROCCZ HCF, PROV MAX ACK M5217 Haddine M, ROCCZ HCF, PROV MAX ACK	#dofine M_HCCC1_HCE_DD0_UN_ACK		M5104
Hardine B, ROCCI RCF, POL IN, ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M519 Haddine M, ROCCZ NCF, ENST, COS ACK M5200 Haddine M, ROCCZ NCF, ENST, COS ACK M5201 Haddine M, ROCCZ NCF, ENST, COS ACK M5203 Haddine M, ROCCZ NCF, ENST, COS ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5203 Haddine M, ROCCZ NCF, PROV MAX ACK M5204 Haddine M, ROCCZ NCF, PROV MAX ACK M5206 Haddine M, ROCCZ NCF, PROV MAX ACK M5207 Haddine M, ROCCZ HCF, BKNM, MAX ACK M5210 Haddine M, ROCCZ HCF, RSNM, MAX ACK M5211 Haddine M, ROCCZ HCF, PROV MAX ACK M5212 Haddine M, ROCCZ HCF, PROV MAX ACK M5213 Haddine M, ROCCZ HCF, PROV MAX ACK M5214 Haddine M, ROCCZ HCF, PROV MAX ACK M5215 Haddine M, ROCCZ HCF, PROV MAX ACK M5216 Haddine M, ROCCZ HCF, PROV MAX ACK M5217 Haddine M, ROCCZ HCF, PROV MAX ACK	#define M_HCCC1_HCF_PD0_IN_ACK		MJ194
idefine M_ECC2_DusNetXrlags M5197 idefine M_ECC2_NCC_CONTONCATING M5198 idefine M_ECC2_NCC_SONT_COS_ACK M5200 idefine M_ECC2_NCC_SONT_COS_ACK M5201 idefine M_ECC2_NCC_SONT_MAX_ACK M5202 idefine M_ECC2_NCC_SONT_MAX_ACK M5204 idefine M_ECC2_NCC_SONT_MAX_ACK M5211 idefine M_ECC2_NCC_SONT_MAX_ACK M5212 idefine M_ECC2_NCC_SONT_MAX_ACK M5214 idefine M_ECC2_NCC_SONT_MAX_ACK M5221 idefine M_ECCC3_NCC_SONT_	#define M_HCCC1_HCF_PD1_OUT_CMD		M3195
#define M_RCC2_NCF_RECV_MEX_CKD MS201 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS204 #define M_RCC2_NCF_PD0_INT_ACK MS205 #define M_RCC2_NCF_PD0_INT_ACK MS206 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS211 #define M_RCC2_NCF_PD0_INT_ACK MS213 #define M_RCC2_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PNTX_COS_ACK MS216 #define M_RCC3_NCF_PNTX_COS_ACK MS218 #define M_RCC3_NCF_PNTX_COS_ACK MS220 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK	#define M_HCCC1_HCF_PD1_IN_ACK		M5196
#define M_RCC2_NCF_RECV_MEX_CKD MS201 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS204 #define M_RCC2_NCF_PD0_INT_ACK MS205 #define M_RCC2_NCF_PD0_INT_ACK MS206 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS211 #define M_RCC2_NCF_PD0_INT_ACK MS213 #define M_RCC2_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PNTX_COS_ACK MS216 #define M_RCC3_NCF_PNTX_COS_ACK MS218 #define M_RCC3_NCF_PNTX_COS_ACK MS220 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK	#define M_HCCC2_usNetxFlags	M5197	
#define M_RCC2_NCF_RECV_MEX_CKD MS201 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS204 #define M_RCC2_NCF_PD0_INT_ACK MS205 #define M_RCC2_NCF_PD0_INT_ACK MS206 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS211 #define M_RCC2_NCF_PD0_INT_ACK MS213 #define M_RCC2_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PNTX_COS_ACK MS216 #define M_RCC3_NCF_PNTX_COS_ACK MS218 #define M_RCC3_NCF_PNTX_COS_ACK MS220 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK	#define M HCCC2 NCF COMMUNICATING		M5198
#define M_RCC2_NCF_RECV_MEX_CKD MS201 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS204 #define M_RCC2_NCF_PD0_INT_ACK MS205 #define M_RCC2_NCF_PD0_INT_ACK MS206 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS211 #define M_RCC2_NCF_PD0_INT_ACK MS213 #define M_RCC2_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PNTX_COS_ACK MS216 #define M_RCC3_NCF_PNTX_COS_ACK MS218 #define M_RCC3_NCF_PNTX_COS_ACK MS220 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK	#define M HCCC2 NCF ERROR	M5199	
#define M_RCC2_NCF_RECV_MEX_CKD MS201 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS203 #define M_RCC2_NCF_RECV_MEX_CKD MS204 #define M_RCC2_NCF_PD0_INT_ACK MS205 #define M_RCC2_NCF_PD0_INT_ACK MS206 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD1_IN_CMD MS207 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS210 #define M_RCC2_NCF_PD0_INT_ACK MS211 #define M_RCC2_NCF_PD0_INT_ACK MS213 #define M_RCC2_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PD0_INT_ACK MS215 #define M_RCC3_NCF_PNTX_COS_ACK MS216 #define M_RCC3_NCF_PNTX_COS_ACK MS218 #define M_RCC3_NCF_PNTX_COS_ACK MS220 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PNTX_COS_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK MS223 #define M_RCC3_NCF_PD0_INT_ACK	#define M HCCC2 NCF HOST COS ACK		M5200
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>#define M HCCC2 NCF NETX COS CMD</td> <td></td> <td>M5201</td>	#define M HCCC2 NCF NETX COS CMD		M5201
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>#define M HCCC2 NCE SEND MBX ACK</td> <td></td> <td></td>	#define M HCCC2 NCE SEND MBX ACK		
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>#define M HCCC2 NCE DECV MPX CMD</td> <td></td> <td>M5203</td>	#define M HCCC2 NCE DECV MPX CMD		M5203
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>HARFING MINIGOOD NOT DOO OWE ACT</td> <td></td> <td>ME 204</td>	HARFING MINIGOOD NOT DOO OWE ACT		ME 204
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>#define M_HCCC2_NCF_PDU_OUT_ACK</td> <td></td> <td></td>	#define M_HCCC2_NCF_PDU_OUT_ACK		
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>#define M_HCCC2_NCF_PD0_IN_CMD</td> <td></td> <td>M52U5</td>	#define M_HCCC2_NCF_PD0_IN_CMD		M52U5
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>#define M_HCCC2_NCF_PD1_OUT_ACK</td> <td></td> <td>M5206</td>	#define M_HCCC2_NCF_PD1_OUT_ACK		M5206
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>#define M_HCCC2_NCF_PD1_IN_CMD</td> <td></td> <td>M5207</td>	#define M_HCCC2_NCF_PD1_IN_CMD		M5207
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>#define M HCCC2 usHostFlags</td> <td>M5208</td> <td></td>	#define M HCCC2 usHostFlags	M5208	
Wdefine W. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_RECV_MBX_ACK M5211 #define M. HCCC2_HCF_PDO_OUT_CMD M5213 #define M. HCCC2_HCF_PDO_IN_ACK M5214 #define M. HCCC2_HCF_PDI_UN_ACK M5215 #define M. HCCC3_HCF_PDI_UN_ACK M5216 #define M. HCCC3_UNEC_COMMUNICATING M5218 #define M. HCCC3_NCF_ERROR M5219 #define M. HCCC3_NCF_ERROR M5221 #define M. HCCC3_NCF_ERROR M5220 #define M. HCCC3_NCF_POST_COS_ACK M5222 #define M. HCCC3_NCF_POD_UN_CM M5223 #define M. HCCC3_NCF_POD_UN_CM M5224 #define M. HCCC3_NCF_POD_UN_CM M5226 #define M. HCCC3_NCF_POD_UN_CM M5227 #define M. HCCC3_NCF_NOS_COS_ACK M5230 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_NOS_COS_ACK M5231 #define M. HCCC3_NCF_POD_UN_CM M5232 <td>#define M HCCC2 HCF HOST COS CMD</td> <td></td> <td></td>	#define M HCCC2 HCF HOST COS CMD		
#define M_HCC3_USNetxPlagsM5217#define M_HCC3_NCF_COMMUNICATINGM5219#define M_HCC3_NCF_ERBORM5219#define M_HCC3_NCF_NETX_COS_CMDM5221#define M_HCC3_NCF_SEDV_MSX_CAKM5222#define M_HCC3_NCF_PD0_OUT_ACKM5223#define M_HCC3_NCF_PD1_NCMDM5225#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5227#define M_HCC3_NCF_PD1_NCMDM5228#define M_HCC3_NCF_PD1_NCMDM5229#define M_HCC3_NCF_PD1_NCMDM5231#define M_HCC3_NCF_PD1_OUT_CMDM5231#define M_HCC3_HCF_PD0_OUT_CMDM5233#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_NACKM5236#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5237#define M_HCC0_NCF_CREVCNMUNICATINGM5237#define M_HCAC0_NCF_CREVCNMUNICATINGM5238#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5241#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_NCMDM5243<	#define M HCCC2 HCF NETX COS ACK		
#define M_HCC3_USNetxPlagsM5217#define M_HCC3_NCF_COMMUNICATINGM5219#define M_HCC3_NCF_ERBORM5219#define M_HCC3_NCF_NETX_COS_CMDM5221#define M_HCC3_NCF_SEDV_MSX_CAKM5222#define M_HCC3_NCF_PD0_OUT_ACKM5223#define M_HCC3_NCF_PD1_NCMDM5225#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5227#define M_HCC3_NCF_PD1_NCMDM5228#define M_HCC3_NCF_PD1_NCMDM5229#define M_HCC3_NCF_PD1_NCMDM5231#define M_HCC3_NCF_PD1_OUT_CMDM5231#define M_HCC3_HCF_PD0_OUT_CMDM5233#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_NACKM5236#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5237#define M_HCC0_NCF_CREVCNMUNICATINGM5237#define M_HCAC0_NCF_CREVCNMUNICATINGM5238#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5241#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_NCMDM5243<	#define M HCCC2 HCF SEND MBY CMD		 M5211
#define M_HCC3_USNetxPlagsM5217#define M_HCC3_NCF_COMMUNICATINGM5219#define M_HCC3_NCF_ERBORM5219#define M_HCC3_NCF_NETX_COS_CMDM5221#define M_HCC3_NCF_SEDV_MSX_CAKM5222#define M_HCC3_NCF_PD0_OUT_ACKM5223#define M_HCC3_NCF_PD1_NCMDM5225#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5227#define M_HCC3_NCF_PD1_NCMDM5228#define M_HCC3_NCF_PD1_NCMDM5229#define M_HCC3_NCF_PD1_NCMDM5231#define M_HCC3_NCF_PD1_OUT_CMDM5231#define M_HCC3_HCF_PD0_OUT_CMDM5233#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_NACKM5236#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5237#define M_HCC0_NCF_CREVCNMUNICATINGM5237#define M_HCAC0_NCF_CREVCNMUNICATINGM5238#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5241#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_NCMDM5243<	#define M HCCC2 HCF DECK MBY ACK		M5212
#define M_HCC3_USNetxPlagsM5217#define M_HCC3_NCF_COMMUNICATINGM5219#define M_HCC3_NCF_ERBORM5219#define M_HCC3_NCF_NETX_COS_CMDM5221#define M_HCC3_NCF_SEDV_MSX_CAKM5222#define M_HCC3_NCF_PD0_OUT_ACKM5223#define M_HCC3_NCF_PD1_NCMDM5225#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5227#define M_HCC3_NCF_PD1_NCMDM5228#define M_HCC3_NCF_PD1_NCMDM5229#define M_HCC3_NCF_PD1_NCMDM5231#define M_HCC3_NCF_PD1_OUT_CMDM5231#define M_HCC3_HCF_PD0_OUT_CMDM5233#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_NACKM5236#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5237#define M_HCC0_NCF_CREVCNMUNICATINGM5237#define M_HCAC0_NCF_CREVCNMUNICATINGM5238#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5241#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_NCMDM5243<	#define M_HCCC2_HCF_KECV_MBA_ACK		M5212
#define M_HCC3_USNetxPlagsM5217#define M_HCC3_NCF_COMMUNICATINGM5219#define M_HCC3_NCF_ERBORM5219#define M_HCC3_NCF_NETX_COS_CMDM5221#define M_HCC3_NCF_SEDV_MSX_CAKM5222#define M_HCC3_NCF_PD0_OUT_ACKM5223#define M_HCC3_NCF_PD1_NCMDM5225#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5227#define M_HCC3_NCF_PD1_NCMDM5228#define M_HCC3_NCF_PD1_NCMDM5229#define M_HCC3_NCF_PD1_NCMDM5231#define M_HCC3_NCF_PD1_OUT_CMDM5231#define M_HCC3_HCF_PD0_OUT_CMDM5233#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_NACKM5236#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5237#define M_HCC0_NCF_CREVCNMUNICATINGM5237#define M_HCAC0_NCF_CREVCNMUNICATINGM5238#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5241#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_NCMDM5243<	#define M_HCCC2_HCF_PDU_OUT_CMD		M5213
#define M_HCC3_USNetxPlagsM5217#define M_HCC3_NCF_COMMUNICATINGM5219#define M_HCC3_NCF_ERBORM5219#define M_HCC3_NCF_NETX_COS_CMDM5221#define M_HCC3_NCF_SEDV_MSX_CAKM5222#define M_HCC3_NCF_PD0_OUT_ACKM5223#define M_HCC3_NCF_PD1_NCMDM5225#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5227#define M_HCC3_NCF_PD1_NCMDM5228#define M_HCC3_NCF_PD1_NCMDM5229#define M_HCC3_NCF_PD1_NCMDM5231#define M_HCC3_NCF_PD1_OUT_CMDM5231#define M_HCC3_HCF_PD0_OUT_CMDM5233#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_NACKM5236#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5237#define M_HCC0_NCF_CREVCNMUNICATINGM5237#define M_HCAC0_NCF_CREVCNMUNICATINGM5238#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5241#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_NCMDM5243<	#define M_HCCC2_HCF_PD0_IN_ACK		
#define M_HCC3_USNetxPlagsM5217#define M_HCC3_NCF_COMMUNICATINGM5219#define M_HCC3_NCF_ERBORM5219#define M_HCC3_NCF_NETX_COS_CMDM5221#define M_HCC3_NCF_SEDV_MSX_CAKM5222#define M_HCC3_NCF_PD0_OUT_ACKM5223#define M_HCC3_NCF_PD1_NCMDM5225#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5227#define M_HCC3_NCF_PD1_NCMDM5228#define M_HCC3_NCF_PD1_NCMDM5229#define M_HCC3_NCF_PD1_NCMDM5231#define M_HCC3_NCF_PD1_OUT_CMDM5231#define M_HCC3_HCF_PD0_OUT_CMDM5233#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_NACKM5236#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5237#define M_HCC0_NCF_CREVCNMUNICATINGM5237#define M_HCAC0_NCF_CREVCNMUNICATINGM5238#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5241#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_NCMDM5243<	#define M_HCCC2_HCF_PD1_OUT_CMD		M5215
#define M_HCC3_USNetxPlagsM5217#define M_HCC3_NCF_COMMUNICATINGM5219#define M_HCC3_NCF_ERBORM5219#define M_HCC3_NCF_NETX_COS_CMDM5221#define M_HCC3_NCF_SEDV_MSX_CAKM5222#define M_HCC3_NCF_PD0_OUT_ACKM5223#define M_HCC3_NCF_PD1_NCMDM5225#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5226#define M_HCC3_NCF_PD1_NCMDM5227#define M_HCC3_NCF_PD1_NCMDM5228#define M_HCC3_NCF_PD1_NCMDM5229#define M_HCC3_NCF_PD1_NCMDM5231#define M_HCC3_NCF_PD1_OUT_CMDM5231#define M_HCC3_HCF_PD0_OUT_CMDM5233#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_NACKM5236#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5235#define M_HCC3_HCF_PD1_OUT_CMDM5237#define M_HCC0_NCF_CREVCNMUNICATINGM5237#define M_HCAC0_NCF_CREVCNMUNICATINGM5238#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_REVCNMUNICATINGM5241#define M_HCAC0_NCF_REVCNMUNICATINGM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5243#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_OUT_ACKM5241#define M_HCAC0_NCF_PD1_NCMDM5243<	#define M_HCCC2_HCF_PD1_IN_ACK		M5216
#define M_HCCC3_NCF_ERROR M5219 #define M_HCCC3_NCF_NETX_COS_CMD M5221 #define M_HCCC3_NCF_SED_MBX_CK M5222 #define M_HCCC3_NCF_SED_MBX_CK M5223 #define M_HCCC3_NCF_PD0_OUT_ACK M5224 #define M_HCCC3_NCF_PD1_N_CMD M5225 #define M_HCCC3_NCF_PD1_N_CMD M5227 #define M_HCCC3_NCF_PD1_N_CMD M5227 #define M_HCCC3_UFF_NETX_COS_CMD M5229 #define M_HCCC3_HCF_NETX_COS_ACK M5231 #define M_HCCC3_HCF_PD1_OUT_CMD M5231 #define M_HCCC3_HCF_PD1_OUT_CMD M5233 #define M_HCCC3_HCF_PD1_N_ACK M5234 #define M_HCCC3_HCF_PD1_N_ACK M5233 #define M_HCCC3_HCF_PD1_N_ACK M5236 #define M_HCCC3_HCF_PD1_N_ACK M5246 #define M_HCCC3_HCF_PD1_N_ACK M5246 #define M_HCCC3_HCF_PD1_N_ACK M5246 #define M_HCCC0_HCF_HOST_COS_ACK M5	#define M HCCC3 usNetxFlags	M5217	
#define M_HCCC3_NCF_ERROR M5219 #define M_HCCC3_NCF_NETX_COS_CMD M5221 #define M_HCCC3_NCF_SED_MBX_CK M5222 #define M_HCCC3_NCF_SED_MBX_CK M5223 #define M_HCCC3_NCF_PD0_OUT_ACK M5224 #define M_HCCC3_NCF_PD1_N_CMD M5225 #define M_HCCC3_NCF_PD1_N_CMD M5227 #define M_HCCC3_NCF_PD1_N_CMD M5227 #define M_HCCC3_UFF_NETX_COS_CMD M5229 #define M_HCCC3_HCF_NETX_COS_ACK M5231 #define M_HCCC3_HCF_PD1_OUT_CMD M5231 #define M_HCCC3_HCF_PD1_OUT_CMD M5233 #define M_HCCC3_HCF_PD1_N_ACK M5234 #define M_HCCC3_HCF_PD1_N_ACK M5233 #define M_HCCC3_HCF_PD1_N_ACK M5236 #define M_HCCC3_HCF_PD1_N_ACK M5246 #define M_HCCC3_HCF_PD1_N_ACK M5246 #define M_HCCC3_HCF_PD1_N_ACK M5246 #define M_HCCC0_HCF_HOST_COS_ACK M5	#define M HCCC3 NCF COMMUNICATING		M5218
#define M_HCCC3_NCT_HOST_COS_ACKM5220#define M_HCCC3_NCT_SEND_MBX_ACKM5221#define M_HCCC3_NCT_SEND_MBX_ACKM5223#define M_HCCC3_NCT_PD0_IN_CMDM5223#define M_HCCC3_NCT_PD1_OUT_ACKM5225#define M_HCCC3_NCT_PD1_OUT_ACKM5226#define M_HCCC3_USHOSTPlagsM5227#define M_HCCC3_USHOSTPlagsM5228#define M_HCCC3_HCT_NET_NCMDM5223#define M_HCCC3_HCT_NET_NCOS_ACKM5230#define M_HCCC3_HCT_NET_NCOS_ACKM5231#define M_HCCC3_HCT_NET_NCOS_ACKM5233#define M_HCCC3_HCT_PD1_OUT_CMDM5233#define M_HCCC3_HCT_PD1_OUT_CMDM5235#define M_HCCC3_HCT_PD1_OUT_CMDM5235#define M_HCCC3_HCT_PD1_OUT_CMDM5236#define M_HCCC3_HCT_PD1_OUT_CMDM5236#define M_HCCC3_HCT_PD1_OUT_CMDM5237#define M_HCCC3_HCT_PD1_OUT_CMDM5236#define M_HCCC3_HCT_PD1_N_ACKM5236#define M_HCCC3_HCT_NET_COS_CMDM5241#define M_HCCC3_HCT_NET_COS_CMDM5243#define M_HCCC0_NCT_NET_COS_CMDM5243#define M_HCCC0_NCT_RECV_MEX_CMDM5243#define M_HCCC0_NCT_RECV_MEX_CMDM5243#define M_HCCC0_NCT_RECV_MEX_CMDM5244#define M_HCCC0_NCT_PD0_UT_ACKM5245#define M_HCCC0_NCT_PD1_NCMDM5246#define M_HCCC0_NCT_PD1_NCMDM5246#define M_HCCC0_NCT_PD1_NCMDM5246#define M_HCCC0_NCT_PD1_NCMDM5246#define M_HCCC0_NCT_PD1_NCMDM5246#define M_HCCC0_NCT_PD1_NCMD <t< td=""><td>#define M HCCC3 NCF ERROR</td><td>M5219</td><td></td></t<>	#define M HCCC3 NCF ERROR	M5219	
#define M_HCCC3_NCF_NETX_COS_CMD M5221 #define M_HCCC3_NCF_RECV_MBX_ACK M5222 #define M_HCCC3_NCF_PD0_OUT_ACK M5223 #define M_HCCC3_NCF_PD0_OUT_ACK M5224 #define M_HCCC3_NCF_PD1_IN_CMD M5225 #define M_HCCC3_NCF_PD1_IN_CMD M5227 #define M_HCCC3_NCF_PD1_IN_CMD M5227 #define M_HCCC3_HCF_HOST_COS_CMD M5229 #define M_HCCC3_HCF_SND_MBX_CMD M5231 #define M_HCCC3_HCF_SND_MBX_CMD M5231 #define M_HCCC3_HCF_PD0_OUT_CMD M5233 #define M_HCCC3_HCF_PD0_OUT_CMD M5235 #define M_HCCC3_HCF_PD1_N_ACK M5236 #define M_HCCC3_HCF_PD1_N_ACK M5236 #define M_HCAC0_NCF_COMMUNICATING M5238 #define M_HCAC0_NCF_FRECV_MBX_CMD M5241 #define M_HCAC0_NCF_FRECV_MBX_CMD M5242 #define M_HCAC0_NCF_FRECV_MBX_CMD M5243 #define M_HCAC0_NCF_FRECV_MBX_CMD M5244 </td <td>#define M HCCC3 NCF HOST COS ACK</td> <td></td> <td>м5220</td>	#define M HCCC3 NCF HOST COS ACK		м5220
#defineM_HCCC3_NCF_PED_MBX_ACKM5222#defineM_HCCC3_NCF_PED_OUT_ACKM5223#defineM_HCCC3_NCF_PED_IN_CMDM5225#defineM_HCCC3_NCF_PED_IN_CMDM5226#defineM_HCCC3_NCF_PED_IN_CMDM5227#defineM_HCCC3_HCF_NETX_COS_ACKM5230#defineM_HCCC3_HCF_NETX_COS_ACKM5231#defineM_HCCC3_HCF_PED_IN_ACKM5233#defineM_HCCC3_HCF_PED_OUT_CMDM5233#defineM_HCCC3_HCF_PED_OUT_CMDM5235#defineM_HCCC3_HCF_PED_IN_ACKM5236#defineM_HCCC3_HCF_PED_IN_ACKM5236#defineM_HCCC3_HCF_PED_IN_ACKM5236#defineM_HCCC3_HCF_PED_IN_ACKM5236#defineM_HCCC3_HCF_PED_IN_ACKM5236#defineM_HCCC0_NCF_COMUNICATINGM5238#defineM_HCAC0_NCF_COMUNICATINGM5240#defineM_HCAC0_NCF_ERCRM5240#defineM_HCAC0_NCF_NEX_COS_CMDM5241#defineM_HCAC0_NCF_PED_UT_ACKM5241#defineM_HCAC0_NCF_PED_UT_ACKM5243#defineM_HCAC0_NCF_PED_UT_ACKM5246#defineM_HCAC0_NCF_PED_UT_ACKM5246#defineM_HCAC0_NCF_PED_UT_ACKM5246#defineM_HCAC0_NCF_PED_UT_ACKM5246#defineM_HCAC0_NCF_PED_NEX_CMDM5247#defineM_HCAC0_NCF_PED_NEX_CMDM5247#defineM_HCAC0_NCF_PED_NEX_CMDM5246#defineM_HCAC0_NCF_PED_NEX_CMDM5246 </td <td>#define M HCCC3 NCE NETY COS CMD</td> <td></td> <td></td>	#define M HCCC3 NCE NETY COS CMD		
#define M_HCCC3_NCF_PDI_IN_CMDM5227#define M_HCCC3_HCF_HOST_COS_CMDM5228#define M_HCCC3_HCF_NETX_COS_ACKM5230#define M_HCCC3_HCF_SEND_MBX_CMDM5231#define M_HCCC3_HCF_RECV_MBX_ACKM5232#define M_HCCC3_HCF_PD0_OUT_CMDM5233#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCAC0_USNEtxFlagsM5237#define M_HCAC0_NCF_COMMUNICATINGM5238#define M_HCAC0_NCF_HOST_COS_ACKM5240#define M_HCAC0_NCF_NETX_COS_CMDM5241#define M_HCAC0_NCF_REV_MBX_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD1_IN_CMDM5245#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_HCF_HOST_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC	#define M_HCCCS_NCF_NEIX_COS_CMD		
#define M_HCCC3_NCF_PDI_IN_CMDM5227#define M_HCCC3_HCF_HOST_COS_CMDM5228#define M_HCCC3_HCF_NETX_COS_ACKM5230#define M_HCCC3_HCF_SEND_MBX_CMDM5231#define M_HCCC3_HCF_RECV_MBX_ACKM5232#define M_HCCC3_HCF_PD0_OUT_CMDM5233#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCAC0_USNEtxFlagsM5237#define M_HCAC0_NCF_COMMUNICATINGM5238#define M_HCAC0_NCF_HOST_COS_ACKM5240#define M_HCAC0_NCF_NETX_COS_CMDM5241#define M_HCAC0_NCF_REV_MBX_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD1_IN_CMDM5245#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_HCF_HOST_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC	#define M_HCCC3_NCF_SEND_MBX_ACK		M5222
#define M_HCCC3_NCF_PDI_IN_CMDM5227#define M_HCCC3_HCF_HOST_COS_CMDM5228#define M_HCCC3_HCF_NETX_COS_ACKM5230#define M_HCCC3_HCF_SEND_MBX_CMDM5231#define M_HCCC3_HCF_RECV_MBX_ACKM5232#define M_HCCC3_HCF_PD0_OUT_CMDM5233#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCAC0_USNEtxFlagsM5237#define M_HCAC0_NCF_COMMUNICATINGM5238#define M_HCAC0_NCF_HOST_COS_ACKM5240#define M_HCAC0_NCF_NETX_COS_CMDM5241#define M_HCAC0_NCF_REV_MBX_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD1_IN_CMDM5245#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_HCF_HOST_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC	#define M_HCCC3_NCF_RECV_MBX_CMD		M5223
#define M_HCCC3_NCF_PDI_IN_CMDM5227#define M_HCCC3_HCF_HOST_COS_CMDM5228#define M_HCCC3_HCF_NETX_COS_ACKM5230#define M_HCCC3_HCF_SEND_MBX_CMDM5231#define M_HCCC3_HCF_RECV_MBX_ACKM5232#define M_HCCC3_HCF_PD0_OUT_CMDM5233#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCAC0_USNEtxFlagsM5237#define M_HCAC0_NCF_COMMUNICATINGM5238#define M_HCAC0_NCF_HOST_COS_ACKM5240#define M_HCAC0_NCF_NETX_COS_CMDM5241#define M_HCAC0_NCF_REV_MBX_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD1_IN_CMDM5245#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_HCF_HOST_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC	#define M_HCCC3_NCF_PD0_OUT_ACK		M5224
#define M_HCCC3_NCF_PDI_IN_CMDM5227#define M_HCCC3_HCF_HOST_COS_CMDM5228#define M_HCCC3_HCF_NETX_COS_ACKM5230#define M_HCCC3_HCF_SEND_MBX_CMDM5231#define M_HCCC3_HCF_RECV_MBX_ACKM5232#define M_HCCC3_HCF_PD0_OUT_CMDM5233#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCAC0_USNEtxFlagsM5237#define M_HCAC0_NCF_COMMUNICATINGM5238#define M_HCAC0_NCF_HOST_COS_ACKM5240#define M_HCAC0_NCF_NETX_COS_CMDM5241#define M_HCAC0_NCF_REV_MBX_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD1_IN_CMDM5245#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_HCF_HOST_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC	#define M_HCCC3_NCF_PD0_IN_CMD		M5225
#define M_HCCC3_NCF_PDI_IN_CMDM5227#define M_HCCC3_HCF_HOST_COS_CMDM5228#define M_HCCC3_HCF_NETX_COS_ACKM5230#define M_HCCC3_HCF_SEND_MBX_CMDM5231#define M_HCCC3_HCF_RECV_MBX_ACKM5232#define M_HCCC3_HCF_PD0_OUT_CMDM5233#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCCC3_HCF_PD1_IN_ACKM5236#define M_HCAC0_USNEtxFlagsM5237#define M_HCAC0_NCF_COMMUNICATINGM5238#define M_HCAC0_NCF_HOST_COS_ACKM5240#define M_HCAC0_NCF_NETX_COS_CMDM5241#define M_HCAC0_NCF_REV_MBX_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD0_OUT_ACKM5243#define M_HCAC0_NCF_PD1_IN_CMDM5245#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_NCF_PD1_IN_CMDM5247#define M_HCAC0_HCF_HOST_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5249#define M_HCAC0_HCF_NETX_COS_CMDM5241#define M_HCAC	#define M HCCC3 NCF PD1 OUT ACK		М5226
#defineM-HCCC3_usHostFlagsM5228#defineM-HCCC3_HCF_HOST_COS_CMDM5230#defineM-HCCC3_HCF_NETX_COS_ACKM5231#defineM-HCCC3_HCF_RECV_MBX_ACKM5231#defineM-HCCC3_HCF_PD0_OUT_CMDM5233#defineM-HCCC3_HCF_PD0_OUT_CMDM5235#defineM-HCCC3_HCF_PD1_OUT_CMDM5235#defineM-HCCC3_HCF_PD1_OUT_CMDM5235#defineM-HCCC3_HCF_PD1_IN_ACKM5236#defineM-HCAC0_usNetxFlagsM5237#defineM-HCAC0_NCF_COMMUNICATINGM5238#defineM-HCAC0_NCF_HOST_COS_ACKM5240#defineM-HCAC0_NCF_HOST_COS_ACKM5241#defineM-HCAC0_NCF_SEND_MBX_ACKM5242#defineM-HCAC0_NCF_PD_OUT_ACKM5243#defineM-HCAC0_NCF_PD_OUT_ACKM5243#defineM-HCAC0_NCF_PD_IN_CMDM5243#defineM-HCAC0_NCF_PD_IN_CMDM5243#defineM-HCAC0_NCF_PD_IN_CMDM5243#defineM-HCAC0_NCF_PD_IN_CMDM5247#defineM-HCAC0_NCF_PD_IN_CMDM5247#defineM-HCAC0_NCF_PD_IN_CMDM5247#defineM-HCAC0_NCF_PD_IN_CMDM5248#defineM-HCAC0_NCF_PD_SCMMM5249#defineM-HCAC0_NCF_PD_SCMMM5249#defineM-HCAC0_HCF_HOST_COS_CMDM5249#defineM-HCAC0_HCF_NSTX_COS_ACKM5250#defineM-HCAC0_HCF_SEND_MEX_CMDM5251	#define M HCCC3 NCF PD1 IN CMD		м5227
#defineM-HCCC3_HCF_HOST_COS_CMDM5229#defineM-HCCC3_HCF_NETX_COS_ACKM5230#defineM-HCCC3_HCF_SEND_MBX_CMDM5231#defineM-HCCC3_HCF_PD0_OUT_CMDM5233#defineM-HCCC3_HCF_PD0_OUT_CMDM5233#defineM-HCCC3_HCF_PD1_OUT_CMDM5235#defineM-HCCC3_HCF_PD1_IN_ACKM5236#defineM-HCCC0_UNF_FD1_IN_ACKM5237#defineM-HCAC0_UNF_FC0MUNICATINGM5237#defineM-HCAC0_NCF_COMMUNICATINGM5238#defineM-HCAC0_NCF_HOST_COS_ACKM5240#defineM-HCAC0_NCF_HOST_COS_ACKM5241#defineM-HCAC0_NCF_SEND_MBX_ACKM5243#defineM-HCAC0_NCF_PD0_UT_ACKM5243#defineM-HCAC0_NCF_PD0_UT_ACKM5244#defineM-HCAC0_NCF_PD0_UT_ACKM5245#defineM-HCAC0_NCF_PD1_UNT_ACKM5246#defineM-HCAC0_USF_FRANCM5247#defineM-HCAC0_USF_HAST_COS_CMDM5247#defineM-HCAC0_USF_HAST_COS_CMDM5247#defineM-HCAC0_USF_HAST_COS_CMDM5249#defineM-HCAC0_USF_SEND_MBX_CMDM5251	#define M HCCC3 usHostFlags	M5228	
#defineM_HCCC3_HCF_NETX_COS_ACKM5230#defineM_HCCC3_HCF_SEND_MBX_CMDM5231#defineM_HCCC3_HCF_PD0_OUT_CMDM5233#defineM_HCCC3_HCF_PD0_IN_ACKM5233#defineM_HCCC3_HCF_PD1_OUT_CMDM5235#defineM_HCCC3_HCF_PD1_IN_ACKM5236#defineM_HCCC0_USNEtxFlagsM5237#defineM_HCAC0_USNEtxFlagsM5237#defineM_HCAC0_NCF_COMMUNICATINGM5238#defineM_HCAC0_NCF_ERRORM5239#defineM_HCAC0_NCF_HOST_COS_ACKM5240#defineM_HCAC0_NCF_SEND_MBX_ACKM5242#defineM_HCAC0_NCF_RETX_COS_CMDM5243#defineM_HCAC0_NCF_PD0_OUT_ACKM5243#defineM_HCAC0_NCF_PD1_OUT_ACKM5245#defineM_HCAC0_NCF_PD1_OUT_ACKM5246#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_USHOST_COS_CMDM5248#defineM_HCAC0_HCF_HOST_COS_CMDM5248#defineM_HCAC0_HCF_HOST_COS_CMDM5248#defineM_HCAC0_HCF_HOST_COS_CMDM5241#defineM_HCAC0_HCF_HOST_COS_CMDM5248#defineM_HCAC0_HCF_HOST_COS_CMDM5240#defineM_HCAC0_HCF_HOST_COS_CMDM5251	#define M HCCC3 HCF HOST COS CMD		M5229
#define M_HCCC3_HCF_SEND_MBX_CMD M5231 #define M_HCCC3_HCF_RECV_MBX_ACK M5232 #define M_HCCC3_HCF_PD0_OUT_CMD M5233 #define M_HCCC3_HCF_PD1_IN_ACK M5235 #define M_HCCC3_HCF_PD1_IN_ACK M5236 #define M_HCCC3_HCF_PD1_IN_ACK M5236 #define M_HCAC0_UNSMEXFLags M5237 #define M_HCAC0_NCF_COMMUNICATING M5238 #define M_HCAC0_NCF_ENROR M5240 #define M_HCAC0_NCF_HOST_COS_CMD M5241 #define M_HCAC0_NCF_NETX_COS_CMD M5241 #define M_HCAC0_NCF_RECV_MBX_ACK M5242 #define M_HCAC0_NCF_RECV_MBX_ACK M5243 #define M_HCAC0_NCF_PD0_OUT_ACK M5243 #define M_HCAC0_NCF_PD0_IN_CMD M5243 #define M_HCAC0_NCF_PD0_IN_CMD M5244 #define M_HCAC0_NCF_PD1_IN_CMD M5247 #define M_HCAC0_NCF_PD1_IN_CMD M5247 #define M_HCAC0_NCF_PD1_IN_CMD M5247 #define M_HCAC0_NCF_PD1_IN_CMD M5247 <	#define M HCCC3 HCF NETY COS ACK		
#defineM_HCCC3_HCF_RECV_MBX_ACKM5232#defineM_HCCC3_HCF_PD0_OUT_CMDM5233#defineM_HCCC3_HCF_PD1_IN_ACKM5235#defineM_HCCC3_HCF_PD1_IN_ACKM5236#defineM_HCCC0_USNetxFlagsM5237#defineM_HCAC0_NCF_COMMUNICATINGM5238#defineM_HCAC0_NCF_ERRORM5239#defineM_HCAC0_NCF_NETX_COS_ACKM5240#defineM_HCAC0_NCF_NETX_COS_CMDM5241#defineM_HCAC0_NCF_RERV MBX_CMDM5242#defineM_HCAC0_NCF_REVMBX_CMDM5243#defineM_HCAC0_NCF_PD0_OUT_ACKM5244#defineM_HCAC0_NCF_PD0_IN_CMDM5245#defineM_HCAC0_NCF_PD1_OUT_ACKM5246#defineM_HCAC0_NCF_PD1_NCMDM5247#defineM_HCAC0_USHSTFLagsM5248#defineM_HCAC0_USHSTFLagsM5248#defineM_HCAC0_HCF_NST_COS_CMDM5249#defineM_HCAC0_HCF_NST_COS_ACKM5240#defineM_HCAC0_HCF_NST_COS_CMDM5249#defineM_HCAC0_HCF_NST_COS_CMDM5241			
#defineM_HCCC3_HCF_PD0_OUT_CMDM5233#defineM_HCCC3_HCF_PD0_IN_ACKM5234#defineM_HCCC3_HCF_PD1_OUT_CMDM5235#defineM_HCCC3_HCF_PD1_IN_ACKM5236#defineM_HCAC0_USNEtxFlagsM5237#defineM_HCAC0_NCF_COMMUNICATINGM5238#defineM_HCAC0_NCF_ERRORM5239#defineM_HCAC0_NCF_HOST_COS_ACKM5240#defineM_HCAC0_NCF_NETX_COS_CMDM5241#defineM_HCAC0_NCF_SEND_MBX_ACKM5242#defineM_HCAC0_NCF_PD0_OUT_ACKM5243#defineM_HCAC0_NCF_PD0_IN_CMDM5245#defineM_HCAC0_NCF_PD1_OUT_ACKM5246#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_USHOSTFlagsM5248#defineM_HCAC0_USHOSTFLAGSM5248#defineM_HCAC0_HCF_HOST_COS_CMDM5245#defineM_HCAC0_HCF_NETX_COS_ACKM5250#defineM_HCAC0_HCF_NETX_COS_ACKM5251			
#defineM_HCCC3_HCF_PD0_IN_ACKM5234#defineM_HCCC3_HCF_PD1_OUT_CMDM5235#defineM_HCCC3_HCF_PD1_IN_ACKM5236#defineM_HCAC0_usNetxFlagsM5237#defineM_HCAC0_NCF_COMMUNICATINGM5238#defineM_HCAC0_NCF_HOST_COS_ACKM5240#defineM_HCAC0_NCF_NETX_COS_CMDM5241#defineM_HCAC0_NCF_SEND_MBX_ACKM5242#defineM_HCAC0_NCF_RECV_MBX_CMDM5243#defineM_HCAC0_NCF_PD0_OUT_ACKM5245#defineM_HCAC0_NCF_PD1_IN_CMDM5246#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_USHSTLagsM5248#defineM_HCAC0_HCF_HOST_COS_CMDM5249#defineM_HCAC0_HCF_NETX_COS_ACKM5240#defineM_HCAC0_HCF_NETX_COS_ACKM5250#defineM_HCAC0_HCF_SEND_MBX_CMDM5251			
#defineM_HCCC3_HCF_PD1_OUT_CMDM5235#defineM_HCCC3_HCF_PD1_IN_ACKM5236#defineM_HCAC0_USNEtxFlagsM5237#defineM_HCAC0_NCF_COMMUNICATINGM5238#defineM_HCAC0_NCF_ERRORM5239#defineM_HCAC0_NCF_NETX_COS_CMDM5241#defineM_HCAC0_NCF_SEND_MBX_ACKM5242#defineM_HCAC0_NCF_RECV_MBX_CMDM5243#defineM_HCAC0_NCF_PD0_OUT_ACKM5244#defineM_HCAC0_NCF_PD1_IN_CMDM5245#defineM_HCAC0_NCF_PD1_IN_CMDM5246#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_NCF_PD1_IN_CMDM5246#defineM_HCAC0_NCF_PD1_IN_CMDM5246#defineM_HCAC0_HCF_PD3_COS_CMDM5249#defineM_HCAC0_HCF_NETX_COS_ACKM5250#defineM_HCAC0_HCF_SEND_MBX_CMDM5251			
#defineM_HCCC3_HCF_PD1_IN_ACKM5236#defineM_HCAC0_usNetxFlagsM5237#defineM_HCAC0_NCF_COMMUNICATINGM5238#defineM_HCAC0_NCF_ERRORM5239#defineM_HCAC0_NCF_HOST_COS_ACKM5241#defineM_HCAC0_NCF_NETX_COS_CMDM5241#defineM_HCAC0_NCF_SEND_MBX_ACKM5242#defineM_HCAC0_NCF_RECV_MBX_CMDM5243#defineM_HCAC0_NCF_PD0_OUT_ACKM5244#defineM_HCAC0_NCF_PD1_UN_CMDM5245#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_HCF_HOST_COS_CMDM5249#defineM_HCAC0_HCF_HOST_COS_CMDM5249#defineM_HCAC0_HCF_NETX_COS_ACKM5250#defineM_HCAC0_HCF_SEND_MBX_CMDM5251			
#defineMHCACO_usNetxFlagsM5237#defineMHCACO_NCF_COMMUNICATINGM5238#defineMHCACO_NCF_ERRORM5239#defineMHCACO_NCF_HOST_COS_ACKM5240#defineMHCACO_NCF_NETX_COS_CMDM5241#defineMHCACO_NCF_SEND_MBX_ACKM5242#defineMHCACO_NCF_RECV_MBX_CMDM5243#defineMHCACO_NCF_PDO_OUT_ACKM5244#defineMHCACO_NCF_PDO_IN_CMDM5245#defineMHCACO_NCF_PDI_OUT_ACKM5245#defineMHCACO_NCF_PDI_OUT_ACKM5247#defineMHCACO_NCF_PDI_IN_CMDM5248#defineMHCACO_URF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_NETX_COS_ACKM5251			M5235
#defineMHCACO_usNetxFlagsM5237#defineMHCACO_NCF_COMMUNICATINGM5238#defineMHCACO_NCF_ERRORM5239#defineMHCACO_NCF_HOST_COS_ACKM5240#defineMHCACO_NCF_NETX_COS_CMDM5241#defineMHCACO_NCF_SEND_MBX_ACKM5242#defineMHCACO_NCF_RECV_MBX_CMDM5243#defineMHCACO_NCF_PDO_OUT_ACKM5244#defineMHCACO_NCF_PDO_IN_CMDM5245#defineMHCACO_NCF_PDI_OUT_ACKM5245#defineMHCACO_NCF_PDI_OUT_ACKM5247#defineMHCACO_NCF_PDI_IN_CMDM5248#defineMHCACO_URF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_NETX_COS_ACKM5251	#define M HCCC3 HCF PD1 IN ACK		м5236
#defineM_HCAC0_NCF_COMMUNICATINGM5238#defineM_HCAC0_NCF_ERRORM5239#defineM_HCAC0_NCF_HOST_COS_ACKM5240#defineM_HCAC0_NCF_NETX_COS_CMDM5241#defineM_HCAC0_NCF_SEND_MBX_ACKM5242#defineM_HCAC0_NCF_RECV_MBX_CMDM5243#defineM_HCAC0_NCF_PD0_OUT_ACKM5244#defineM_HCAC0_NCF_PD0_IN_CMDM5245#defineM_HCAC0_NCF_PD1_OUT_ACKM5245#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_NCF_PD1_IN_CMDM5248#defineM_HCAC0_HCF_HOST_COS_CMDM5249#defineM_HCAC0_HCF_NETX_COS_ACKM5250#defineM_HCAC0_HCF_SEND_MBX_CMDM5251		M5237	
#defineMHCACO_NCF_ERRORM5239#defineMHCACO_NCF_HOST_COS_ACKM5240#defineMHCACO_NCF_NETX_COS_CMDM5241#defineMHCACO_NCF_SEND_MBX_ACKM5242#defineMHCACO_NCF_RECV_MBX_CMDM5243#defineMHCACO_NCF_PD0_OUT_ACKM5244#defineMHCACO_NCF_PD0_IN_CMDM5245#defineMHCACO_NCF_PD1_OUT_ACKM5245#defineMHCACO_NCF_PD1_OUT_ACKM5246#defineMHCACO_NCF_PD1_IN_CMDM5247#defineMHCACO_USHOSTFLAGSM5248#defineMHCACO_HCF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_SEND_MBX_CMDM5251			M5238
#defineM_HCACO_NCF_HOST_COS_ACKM5240#defineM_HCACO_NCF_NETX_COS_CMDM5241#defineM_HCACO_NCF_SEND_MBX_ACKM5242#defineM_HCACO_NCF_RECV_MBX_CMDM5243#defineM_HCACO_NCF_PDO_OUT_ACKM5244#defineM_HCACO_NCF_PDO_IN_CMDM5245#defineM_HCACO_NCF_PDI_OUT_ACKM5246#defineM_HCACO_NCF_PDI_OUT_ACKM5246#defineM_HCACO_NCF_PDI_IN_CMDM5247#defineM_HCACO_NCF_PDI_IN_CMDM5248#defineM_HCACO_HCF_HOST_COS_CMDM5249#defineM_HCACO_HCF_NETX_COS_ACKM5250#defineM_HCACO_HCF_SEND_MBX_CMDM5251		M5230	
#defineM_HCAC0_NCF_NETX_COS_CMDM5241#defineM_HCAC0_NCF_SEND_MBX_ACKM5242#defineM_HCAC0_NCF_RECV_MBX_CMDM5243#defineM_HCAC0_NCF_PD0_OUT_ACKM5244#defineM_HCAC0_NCF_PD0_IN_CMDM5245#defineM_HCAC0_NCF_PD1_OUT_ACKM5246#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_NCF_PD1_IN_CMDM5247#defineM_HCAC0_NCF_PD1_IN_CMDM5248#defineM_HCAC0_HCF_HOST_COS_CMDM5249#defineM_HCAC0_HCF_NETX_COS_ACKM5250#defineM_HCAC0_HCF_SEND_MBX_CMDM5251		115259	M5240
#defineMHCACO_NCF_SEND_MBX_ACKM5242#defineMHCACO_NCF_RECV_MBX_CMDM5243#defineMHCACO_NCF_PDO_OUT_ACKM5244#defineMHCACO_NCF_PDO_IN_CMDM5245#defineMHCACO_NCF_PDI_OUT_ACKM5246#defineMHCACO_NCF_PDI_IN_CMDM5246#defineMHCACO_NCF_PDI_IN_CMDM5247#defineMHCACO_NCF_PDI_IN_CMDM5248#defineMHCACO_USHOSTFlagsM5248#defineMHCACO_HCF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_SEND_MBX_CMDM5251			
<pre>#define M_HCAC0_NCF_RECV_MBX_CMD M5243 #define M_HCAC0_NCF_PD0_OUT_ACK M5244 #define M_HCAC0_NCF_PD0_IN_CMD M5245 #define M_HCAC0_NCF_PD1_OUT_ACK M5246 #define M_HCAC0_NCF_PD1_IN_CMD M5247 #define M_HCAC0_NCF_PD1_IN_CMD M5248 #define M_HCAC0_HCF_HOST_COS_CMD M5249 #define M_HCAC0_HCF_NETX_COS_ACK M5250 #define M_HCAC0_HCF_SEND_MBX_CMD M5251</pre>			
#defineM HCACO_NCF_PDO_OUT_ACKM5244#defineM HCACO_NCF_PDO_IN_CMDM5245#defineM HCACO_NCF_PD1_OUT_ACKM5246#defineM HCACO_NCF_PD1_IN_CMDM5247#defineM HCACO_USHOSTFlagsM5248#defineM HCACO_HCF_HOST_COS_CMDM5249#defineM HCACO_HCF_NETX_COS_ACKM5250#defineM HCACO_HCF_SEND_MBX_CMDM5251			
#defineMHCACO_NCF_PDO_IN_CMDM5245#defineMHCACO_NCF_PD1_OUT_ACKM5246#defineMHCACO_NCF_PD1_IN_CMDM5247#defineMHCACO_usHostFlagsM5248#defineMHCACO_HCF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_SEND_MBX_CMDM5251	#define M HCACO NCE RECV MRX CMD		
#defineMHCACO_NCF_PD1_OUT_ACKM5246#defineMHCACO_NCF_PD1_IN_CMDM5247#defineMHCACO_usHostFlagsM5248#defineMHCACO_HCF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_SEND_MBX_CMDM5251			M5244
#defineMHCACO_NCF_PD1_OUT_ACKM5246#defineMHCACO_NCF_PD1_IN_CMDM5247#defineMHCACO_usHostFlagsM5248#defineMHCACO_HCF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_SEND_MBX_CMDM5251	#define M_HCAC0_NCF_PD0_OUT_ACK		
#defineMHCACO_NCF_PD1_IN_CMDM5247#defineMHCACO_usHostFlagsM5248#defineMHCACO_HCF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_SEND_MBX_CMDM5251	#define M_HCAC0_NCF_PD0_OUT_ACK		M5245
#defineMHCACO_usHostFlagsM5248#defineMHCACO_HCF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_SEND_MBX_CMDM5251	<pre>#define M_HCAC0_NCF_PD0_OUT_ACK #define M_HCAC0_NCF_PD0_IN_CMD</pre>		
#defineMHCACO_HCF_HOST_COS_CMDM5249#defineMHCACO_HCF_NETX_COS_ACKM5250#defineMHCACO_HCF_SEND_MBX_CMDM5251	<pre>#define M_HCACO_NCF_PD0_OUT_ACK #define M_HCACO_NCF_PD0_IN_CMD #define M_HCACO_NCF_PD1_OUT_ACK</pre>		M5246
<pre>#define M_HCAC0_HCF_NETX_COS_ACK M5250 #define M_HCAC0_HCF_SEND_MBX_CMD M5251</pre>	<pre>#define M_HCACO_NCF_PD0_OUT_ACK #define M_HCACO_NCF_PD0_IN_CMD #define M_HCACO_NCF_PD1_OUT_ACK #define M_HCACO_NCF_PD1_IN_CMD</pre>	M5219	M5246
#define M_HCAC0_HCF_SEND_MBX_CMD M5251	<pre>#define M_HCACO_NCF_PD0_OUT_ACK #define M_HCACO_NCF_PD0_IN_CMD #define M_HCACO_NCF_PD1_OUT_ACK #define M_HCACO_NCF_PD1_IN_CMD #define M_HCACO_usHostFlags</pre>	M5248	M5246 M5247
	<pre>#define M_HCACO_NCF_PD0_OUT_ACK #define M_HCACO_NCF_PD0_IN_CMD #define M_HCACO_NCF_PD1_OUT_ACK #define M_HCACO_NCF_PD1_IN_CMD #define M_HCACO_usHostFlags #define M_HCACO_HCF_HOST_COS_CMD</pre>	M5248	M5246 M5247 M5249
#define M_HCAC0_HCF_RECV_MBX_ACK M5252	<pre>#define M_HCACO_NCF_PD0_OUT_ACK #define M_HCACO_NCF_PD0_IN_CMD #define M_HCACO_NCF_PD1_OUT_ACK #define M_HCACO_NCF_PD1_IN_CMD #define M_HCACO_usHostFlags #define M_HCACO_HCF_HOST_COS_CMD #define M_HCACO_HCF_NETX_COS_ACK</pre>	M5248	M5246 M5247 M5249 M5250
	<pre>#define M_HCACO_NCF_PD0_OUT_ACK #define M_HCACO_NCF_PD0_IN_CMD #define M_HCACO_NCF_PD1_OUT_ACK #define M_HCACO_NCF_PD1_IN_CMD #define M_HCACO_USHOStFlags #define M_HCACO_HCF_HOST_COS_CMD #define M_HCACO_HCF_NETX_COS_ACK #define M_HCACO_HCF_SEND_MEX_CMD</pre>	M5248	M5246 M5247 M5249 M5250 M5251

#define M_HCAC0_HCF_PD0_OUT_CMD	M5253	
#define M_HCAC0_HCF_PD0_IN_ACK	M5254	
#define M HCACO HCF PD1 OUT CMD	M5255	
#define M HCACO HCF PD1 IN ACK	M5256	
#define M HCAC1 usNetxFlags M5257		
#define M HCAC1 NCF COMMUNICATING	M5258	
#define M_HCAC1_NCF_ERROR M5259	1152.50	
	M5260	
#define M_HCACI_NCF_HOSI_COS_ACK		
#define M_HCAC1_NCF_NETX_COS_CMD	M5261	
<pre>#define M_HCAC1_NCF_SEND_MBX_ACK #define M_HCAC1_NCF_RECV_MBX_CMD #define M_HCAC1_NCF_PD0_OUT_ACK #define M_HCAC1_NCF_PD0_IN_CMD #define M_HCAC1_NCF_PD0_IN_CMD</pre>	M5262	
#define M_HCAC1_NCF_RECV_MBX_CMD	M5263	
#define M_HCAC1_NCF_PD0_OUT_ACK	M5264	
#define M HCAC1 NCF PD0 IN CMD	M5265	
#define M_HCAC1_NCF_PD1_OUT_ACK	M5266	
#define M HCAC1 NCF PD1 IN CMD	M5267	
#define M HCAC1 usHostFlags M5268		
#define M HCAC1 HCF HOST COS CMD	M5269	
#define M HCAC1 HCF NETX COS ACK	M5270	
#define M_HCAC1_HCE_SEND_MBX_CMD	M5270	
#dofino M HCACL HOE DECK MDY ACK	MEOTO MEOTO	
HUELTHE M HOROL HOF RELV MBX ACK	ME 272	
#define M_HCAC1_HCF_PDU_OUT_CMD	MDZ/3	
#define M_HCAC1_HCF_PD0_IN_ACK	M5274	
#define M_HCAC1_HCF_PD1_OUT_CMD	M5275	
<pre>#define M_HCAC1_NCF_PD1_IN_CMD #define M_HCAC1_usHostFlags M5268 #define M_HCAC1_HCF_HOST_COS_CMD #define M_HCAC1_HCF_NETX_COS_ACK #define M_HCAC1_HCF_SEND_MBX_CMD #define M_HCAC1_HCF_RECV_MBX_ACK #define M_HCAC1_HCF_PD0_OUT_CMD #define M_HCAC1_HCF_PD0_IN_ACK #define M_HCAC1_HCF_PD1_OUT_CMD #define M_HCAC1_HCF_PD1_IN_ACK #define M_HCAC1_HCF_PD1_IN_ACK #define M_CC0_RCX_APP_COS_APP_READY</pre>	M5276	
#define M_CC0_RCX_APP_COS_APP_READY	M5277	
#define M CCO RCX APP COS BUS ON	M5278	
#define M CCO RCX APP COS BUS ON ENABLE	M5280	M5279
#define M CCO RCX APP COS INIT	M5280	
	M5281	
#define M CCO RCX APP COS LOCK CFG	M5282	
		M5283
#define M_CCO_RCX_APP_COS_LOCK_CFG_ENA		MJZ05
#define M_CC0_RCX_APP_COS_DMA M5284		
<pre>#define M_CC0_RCX_APP_COS_DMA_ENABLE</pre>		
#define M_CC0_ulDeviceWatchdog	M5286	
	M5287	
#define M_CC0_RCX_COMM_COS_RUN	M5288	
#define M CCO RCX COMM COS BUS ON	M5288 M5289	
		M5290
#define M_CC0_RCX_COMM_COS_CONFIG_NEW	M5291	
#define M CCO BCX COMM COS BESTART BEO		M5292
<pre>#define M_CC0_RCX_COMM_COS_RESTART_REQ #define M_CC0_RCX_COMM_C0_REQ_ENA</pre>	M5293	
Halafina M CCO DCV CONM COC DMA	MEDOA	
<pre>#define M_CC0_RCX_COMM_COS_DMA #define M_CC0_ulCommunicationState</pre>	MJZ94	
#deline M_CCU_ulCommunicationState	M5294 M5295	
#deline M_CCU_ulCommunicationError	M5296	
#define M_CC0_usVersion M5297		
#define M_CC0_usVersionM5297#define M_CC0_usWatchdogTimeM5298#define M_CC0_bPDInHskModeM5299		
#define M_CC0_bPDInHskMode M5299		
#define M_CCO_bPDInHskMode M5299 #define M_CCO_bPDInSource M5300		
#define M_CC0_bPDOutHskMode M5301		
#define M CCO bPDOutSource M5302		
#define M CCO ulHostWatchdog M5303		
#define M CCO ulErrorCount M5304		
#define M_CCO_bErrorPDInCnt M5306		
#define M_CCO_bErrorPDOutCnt M5307		
#define M_CC0_bErrorSyncCnt M5308		
#define M_CC0_bSyncHskMode M5309		
#define M_CC0_bSyncSource M5310		
<pre>#define M_CC0_bSyncSource M5310 #define M_CC0_ulSlaveState M5311</pre>		
#define M_CCO_ulSlaveState M5311	м5312	
<pre>#define M_CCO_ulSlaveState M5311 #define M_CCO_ulSlaveErrLogInd</pre>		
<pre>#define M_CCO_ulSlaveState M5311 #define M_CCO_ulSlaveErrLogInd #define M_CCO_ulNumOfConfigSlaves</pre>	M5313	
<pre>#define M_CCO_ulSlaveState M5311 #define M_CCO_ulSlaveErrLogInd</pre>		

Next file:

// M-VariableDefinition \$6C000.pmc CLOSE END GAT DEL GAT #Include "MacroNameDefinition \$6C000.h" M_SI_abCookie_0_->Y:\$6C000,0,8 M_SI_abCookie_1_->Y:\$6C000,8,8
M_SI_abCookie_2_->X:\$6C000,0,8
M_SI_abCookie_3_->X:\$6C000,8,8 M SI ulDpmTotalSize->DP:\$6C001 M_SI_ulDeviceNumber->DP:\$6C002 M SI ulSerialNumber->DP:\$6C003 M_SI_ausHwOptions_0_->Y:\$6C004,0,16 M SI_ausHwOptions 1 ->X:\$6C004,0,16
M_SI_ausHwOptions 2 ->Y:\$6C005,0,16
M_SI_ausHwOptions 3 ->X:\$6C005,0,16 M SI usManufacturer->Y:\$6C006,0,16 M_SI_usProductionDate->X:\$6C006,0,16 M SI ulLicenseFlags1->DP:\$6C007 M SI ulLicenseFlags2->DP:\$6C008 M_SI_usNetxLicenseID->Y:\$6C009,0,16 M SI usNetxLicenseFlags->X:\$6C009,0,16 M_SI_usDeviceClass->Y:\$6C00A,0,16 M SI bHwRevision->X:\$6C00A,0,8 M SI bHwCompatibility->X:\$6C00A,8,8 M SI bDevIdNumber->Y:\$6C00B,0,8 M SCI bChannelType->Y:\$6C00C,0,8 M SCI bSizePositionOfHandshake->X:\$6C00C,0,8 M SCI bNumberOfBlocks->X:\$6C00C,8,8 M_SCI_ulSizeOfChannel->DP:\$6C00D M SCI usSizeOfMailbox->Y:\$6C00E,0,16 M_SCI_usMailboxStartOffset->X:\$6C00E,0,16 M HCI bChannelType->Y:\$6C010,0,8 M HCI ulSizeOfChannel->DP:\$6C011 M CCOI bChannelType->Y:\$6C014,0,8 M CCOI bChannelId->Y:\$6C014,8,8 M CCOI bSizePositionOfHandshake->X:\$6C014,0,8 M CC0I bNumberOfBlocks->X:\$6C014,8,8 M_CC0I_ulSizeOfChannel->DP:\$6C015 M CCOI usCommunicationClass->Y:\$6C016,0,16 M CCOI usProtocolClass->X:\$6C016,0,16 M CCOI usConformanceClass->Y:\$6C017,0,16 M_CC1I_bChannelType->Y:\$6C018,0,8 M_CC1I_bChannelId->Y:\$6C018,8,8 M CC1I bSizePositionOfHandshake->X:\$6C018,0,8 M CC1I bNumberOfBlocks->X:\$6C018,8,8 M CC1I ulSizeOfChannel->DP:\$6C019 M CC1I usCommunicationClass->Y:\$6C01A,0,16 M CC1I usProtocolClass->X:\$6C01A,0,16 M CC1I usConformanceClass->Y:\$6C01B,0,16 M CC2I bChannelType->Y:\$6C01C,0,8 M CC2I bChannelId->Y:\$6C01C,8,8 M CC2I bSizePositionOfHandshake->X:\$6C01C,0,8 M CC2I bNumberOfBlocks->X:\$6C01C,8,8 M CC2I ulSizeOfChannel->DP:\$6C01D M CC2I usCommunicationClass->Y:\$6C01E,0,16 M CC2I usProtocolClass->X:\$6C01E,0,16 M CC2I usConformanceClass->Y:\$6C01F,0,16 M CC3I bChannelType->Y:\$6C020,0,8 M CC3I bChannelId->Y:\$6C020,8,8 M CC3I bSizePositionOfHandshake->X:\$6C020,0,8 M CC3I bNumberOfBlocks->X:\$6C020,8,8 M CC3I ulSizeOfChannel->DP:\$6C021 M_CC3I_usCommunicationClass->Y:\$6C022,0,16 M CC3I usProtocolClass->X:\$6C022,0,16 M CC3I usConformanceClass->Y:\$6C023,0,16 M ACOI bChannelType->Y:\$6C024,0,8 M ACOI bChannelId->Y:\$6C024,8,8

M ACOI bSizePositionOfHandshake->X:\$6C024,0,8 M ACOI bNumberOfBlocks->X:\$6C024,8,8 M_AC0I_ulSizeOfChannel->DP:\$6C025 M AC1I bChannelType->Y:\$6C028,0,8 M AC1I bChannelId->Y:\$6C028,8,8 M AC1I bSizePositionOfHandshake->X:\$6C028,0,8 M_AC1I_bNumberOfBlocks->X:\$6C028,8,8 M_AC1I_ulSizeOfChannel->DP:\$6C029 M SCtrl ulSystemCommandCOS->DP:\$6C02E M_SStat_ulSystemCOS->DP:\$6C030 M SStat ulSystemStatus->DP:\$6C031 M SStat ulSystemError->DP:\$6C032 M_SStat_ulBootError->DP:\$6C033 M SStat ulTimeSinceStart->DP:\$6C034 M SStat usCpuLoad->Y:\$6C035,0,16 M SStat ulHWFeatures->DP:\$6C036 M_SSMB_usPackagesAccepted->Y:\$6C040,0,16 M SSMB ulDest->DP:\$6C041 M SSMB ulSrc->DP:\$6C042 M SSMB ulDestId->DP:\$6C043 M SSMB ulSrcId->DP:\$6C044 M_SSMB_ullen->DP:\$6C045 M SSMB ulId->DP:\$6C046 M_SSMB_ulState->DP:\$6C047 M SSMB ulCmd->DP:\$6C048 M SSMB ulExt->DP:\$6C049 M SSMB ulRout->DP:\$6C04A M_SSMB_ultData0->DP:\$6C04B M SSMB ultData1->DP:\$6C04C M SSMB ultData2->DP:\$6C04D M SSMB ultData3->DP:\$6C04E M SSMB ultData4->DP:\$6C04F M_SSMB_ultData5->DP:\$6C050 M SSMB ultData6->DP:\$6C051 M SSMB ultData7->DP:\$6C052 M SSMB ultData8->DP:\$6C053 M SSMB ultData9->DP:\$6C054 M_SSMB_ultData10->DP:\$6C055 M SSMB ultData11->DP:\$6C056 M SSMB ultData12->DP:\$6C057 M SSMB ultData13->DP:\$6C058 M SSMB ultData14->DP:\$6C059 M SSMB ultData15->DP:\$6C05A M SSMB ultData16->DP:\$6C05B M SSMB ultData17->DP:\$6C05C M SSMB ultData18->DP:\$6C05D M SSMB ultData19->DP:\$6C05E M SSMB ultData20->DP:\$6C05F M_SRMB_usWaitingPackages->Y:\$6C060,0,16 M SRMB ulDest->DP:\$6C061 M SRMB ulSrc->DP:\$6C062 M_SRMB_ulDestId->DP:\$6C063 M SRMB ulSrcId->DP:\$6C064 M_SRMB_ullen->DP:\$6C065 M SRMB ulid->DP:\$6C066 M_SRMB_ulState->DP:\$6C067 M SRMB ulCmd->DP:\$6C068 M SRMB ulExt->DP:\$6C069 M SRMB ulRout->DP:\$6C06A M SRMB ultData0->DP:\$6C06B M SRMB ultData1->DP:\$6C06C M SRMB ultData2->DP:\$6C06D M_SRMB_ultData3->DP:\$6C06E M SRMB ultData4->DP:\$6C06F M SRMB ultData5->DP:\$6C070 M SRMB ultData6->DP:\$6C071 M SRMB ultData7->DP:\$6C072 M_SRMB_ultData8->DP:\$6C073 M SRMB ultData9->DP:\$6C074 M_SRMB_ultData10->DP:\$6C075 M SRMB ultData11->DP:\$6C076

M SRMB ultData12->DP:\$6C077
M SRMB ultData13->DP:\$6C078
M_SRMB_ultData14->DP:\$6C079
M SRMB ultData15->DP:\$6C07A
M_SRMB_ultData16->DP:\$6C07B
M SRMB ultData17->DP:\$6C07C
M_SRMB_ultData18->DP:\$6C07D
M SRMB ultData19->DP:\$6C07E
M SRMB ultData20->DP:\$6C07F
<pre>M_HCSC_bNetxFlags->X:\$6C080,0,8</pre>
M HCSC NSF READY->X:\$6C080,0,1
M HCSC NSF ERROR->X:\$6C080,1,1
M_HCSC_NSF_HOST_COS_ACK->X:\$6C080,2,1
M HCSC NSF NETX COS CMD->X:\$6C080,3,1
M_HCSC_NSF_SEND_MBX_ACK->X:\$6C080,4,1
M_HCSC_NSF_RECV_MBX_CMD->X:\$6C080,5,1
M HCSC bHostFlags->X:\$6C080,8,8
M HCSC HSF RESET->X:\$6C080,8,1
M_HCSC_HSF_BOOTSTART->X:\$6C080,9,1
M HCSC HSF HOST COS CMD->X:\$6C080,10,1
M HCSC HSF NETX COS ACK->X:\$6C080,11,1
M HCSC HSF SEND MBX CMD->X:\$6C080,12,1
<pre>M_HCSC_HSF_RECV_MBX_ACK->X:\$6C080,13,1</pre>
M HCCCO usNetxFlags->Y:\$6C082,0,16
M HCCC0 NCF COMMUNICATING->Y:\$6C082,0,1
<pre>M_HCCC0_NCF_ERROR->Y:\$6C082,1,1</pre>
M HCCCO NCF HOST COS ACK->Y:\$6C082,2,1
M HCCC0 NCF NETX COS CMD->Y:\$6C082,3,1
``
M_HCCC0_NCF_SEND_MBX_ACK->Y:\$6C082,4,1
M_HCCC0_NCF_RECV_MBX_CMD->Y:\$6C082,5,1
M HCCCO NCF PDO OUT ACK->Y:\$6C082,6,1
M HCCCO NCF PDO IN CMD->Y:\$6C082,7,1
<pre>M_HCCC0_NCF_PD1_OUT_ACK->Y:\$6C082,8,1</pre>
M_HCCC0_NCF_PD1_IN_CMD->Y:\$6C082,9,1
M HCCCO usHostFlags->X:\$6C082,0,16
M HCCCO HCF HOST COS CMD->X:\$6C082,2,1
M HCCCO HCF NETX COS ACK->X:\$6C082,3,1
<pre>M_HCCC0_HCF_SEND_MBX_CMD->X:\$6C082,4,1</pre>
M HCCCO HCF RECV MBX ACK->X:\$6C082,5,1
M HCCCO HCF PDO OUT CMD->X:\$6C082,6,1
M HCCCO HCF PDO IN ACK->X:\$6C082,7,1
M_HCCC0_HCF_PD1_OUT_CMD->X:\$6C082,8,1
<pre>M_HCCC0_HCF_PD1_IN_ACK->X:\$6C082,9,1</pre>
M HCCC1 usNetxFlags->Y:\$6C083,0,16
M HCCC1 NCF COMMUNICATING->Y:\$6C083,0,1
M HCCC1 NCF ERROR->Y:\$6C083,1,1
<pre>M_HCCC1_NCF_HOST_COS_ACK->Y:\$6C083,2,1</pre>
M HCCC1 NCF NETX COS CMD->Y:\$6C083,3,1
M HCCC1 NCF SEND MBX ACK->Y:\$6C083,4,1
M HCCC1 NCF RECV MBX CMD->Y:\$6C083,5,1
M_HCCC1_NCF_PD0_OUT_ACK->Y:\$6C083,6,1
M HCCC1 NCF PD0 IN CMD->Y:\$6C083,7,1
M HCCC1 NCF PD1 OUT ACK->Y:\$6C083,8,1
M_HCCC1_NCF_PD1_IN_CMD->Y:\$6C083,9,1
M_HCCC1_usHostFlags->X:\$6C083,0,16
M HCCC1 HCF HOST COS CMD->X:\$6C083,2,1
M HCCC1 HCF NETX COS ACK->X:\$6C083,3,1
M_HCCC1_HCF_SEND_MBX_CMD->X:\$6C083,4,1
M_HCCC1_HCF_RECV_MBX_ACK->X:\$6C083,5,1
M HCCC1 HCF PD0 OUT CMD->X:\$6C083,6,1
``
M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1
M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1
M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1
M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1 M_HCCC2_usNetxFlags->Y:\$6C084,0,16
M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1
<pre>M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1 M_HCCC2_uSNetxFlags->Y:\$6C084,0,16 M_HCCC2_NCF_COMMUNICATING->Y:\$6C084,0,1</pre>
<pre>M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1 M_HCCC2_usNetxFlags->Y:\$6C084,0,16 M_HCCC2_NCF_COMMUNICATING->Y:\$6C084,0,1 M_HCCC2_NCF_ERROR->Y:\$6C084,1,1</pre>
<pre>M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1 M_HCCC2_usNetxFlags->Y:\$6C084,0,16 M_HCCC2_NCF_COMMUNICATING->Y:\$6C084,0,1 M_HCCC2_NCF_ERROR->Y:\$6C084,1,1 M_HCCC2_NCF_HOST_COS_ACK->Y:\$6C084,2,1</pre>
<pre>M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1 M_HCCC2_uSNetxFlags->Y:\$6C084,0,16 M_HCCC2_NCF_COMMUNICATING->Y:\$6C084,0,1 M_HCCC2_NCF_ERROR->Y:\$6C084,1,1 M_HCCC2_NCF_HOST_COS_ACK->Y:\$6C084,2,1 M_HCCC2_NCF_NETX_COS_CMD->Y:\$6C084,3,1</pre>
<pre>M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1 M_HCCC2_usNetxFlags->Y:\$6C084,0,16 M_HCCC2_NCF_COMMUNICATING->Y:\$6C084,0,1 M_HCCC2_NCF_ERROR->Y:\$6C084,1,1 M_HCCC2_NCF_HOST_COS_ACK->Y:\$6C084,2,1</pre>
<pre>M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1 M_HCCC2_uSNetxFlags->Y:\$6C084,0,16 M_HCCC2_NCF_COMMUNICATING->Y:\$6C084,0,1 M_HCCC2_NCF_ERROR->Y:\$6C084,1,1 M_HCCC2_NCF_HOST_COS_ACK->Y:\$6C084,2,1 M_HCCC2_NCF_NETX_COS_CMD->Y:\$6C084,3,1</pre>
<pre>M_HCCC1_HCF_PD0_IN_ACK->X:\$6C083,7,1 M_HCCC1_HCF_PD1_OUT_CMD->X:\$6C083,8,1 M_HCCC1_HCF_PD1_IN_ACK->X:\$6C083,9,1 M_HCCC2_uSNetxFlags->Y:\$6C084,0,16 M_HCCC2_NCF_COMMUNICATING->Y:\$6C084,0,1 M_HCCC2_NCF_ERROR->Y:\$6C084,1,1 M_HCCC2_NCF_HOST_COS_ACK->Y:\$6C084,2,1 M_HCCC2_NCF_NETX_COS_CMD->Y:\$6C084,3,1 M_HCCC2_NCF_SEND_MBX_ACK->Y:\$6C084,4,1</pre>

Appendix A – Setup Examples

M_HCCC2_NCF_PD0_IN_CMD->Y:\$6C084,7,1 M HCCC2 NCF PD1 OUT ACK->Y:\$6C084,8,1 M_HCCC2_NCF_PD1_IN_CMD->Y:\$6C084,9,1 M HCCC2 usHostFlags->X:\$6C084,0,16 M HCCC2 HCF HOST COS CMD->X:\$6C084,2,1 M HCCC2 HCF NETX COS ACK->X:\$6C084,3,1 M_HCCC2_HCF_SEND_MBX_CMD->X:\$6C084,4,1 M_HCCC2_HCF_RECV_MBX_ACK->X:\$6C084,5,1 M HCCC2 HCF PD0 OUT CMD->X:\$6C084,6,1 M_HCCC2_HCF_PD0_IN_ACK->X:\$6C084,7,1 M_HCCC2_HCF_PD1_OUT_CMD->X:\$6C084,8,1 M_HCCC2_HCF_PD1_IN_ACK->X:\$6C084,9,1 M HCCC3 usNetxFlags->Y:\$6C085,0,16 M HCCC3 NCF COMMUNICATING->Y:\$6C085,0,1 M HCCC3 NCF ERROR->Y:\$6C085,1,1 M HCCC3 NCF HOST COS ACK->Y:\$6C085,2,1 M_HCCC3_NCF_NETX_COS_CMD->Y:\$6C085,3,1 M HCCC3 NCF SEND MBX ACK->Y:\$6C085,4,1 M HCCC3 NCF RECV MBX CMD->Y:\$6C085,5,1 M HCCC3 NCF PD0 OUT ACK->Y:\$6C085,6,1 M_HCCC3_NCF_PD0_IN_CMD->Y:\$6C085,7,1 M_HCCC3_NCF_PD1_OUT_ACK->Y:\$6C085,8,1 M HCCC3 NCF PD1 IN CMD->Y:\$6C085,9,1 M_HCCC3_usHostFlags->X:\$6C085,0,16 M HCCC3 HCF HOST COS CMD->X:\$6C085,2,1 M HCCC3 HCF NETX COS ACK->X:\$6C085,3,1 M HCCC3 HCF SEND MBX CMD->X:\$6C085,4,1 M HCCC3 HCF RECV MBX ACK->x:\$6C085,5,1
M HCCC3 HCF PD0 OUT CMD->x:\$6C085,6,1 M HCCC3 HCF PD0 IN ACK->X:\$6C085,7,1 M_HCCC3_HCF_PD1_OUT_CMD->X:\$6C085,8,1 M HCCC3 HCF PD1 IN ACK->X:\$6C085,9,1 M_HCAC0_usNetxFlags->Y:\$6C086,0,16 M HCAC0 NCF COMMUNICATING->Y:\$6C086,0,1 M HCAC0_NCF_ERROR->Y:\$6C086,1,1
M_HCAC0_NCF_HOST_COS_ACK->Y:\$6C086,2,1 M HCACO NCF NETX COS CMD->Y:\$6C086,3,1 M_HCAC0_NCF_SEND_MBX_ACK->Y:\$6C086,4,1 M HCACO NCF RECV MBX CMD->Y:\$6C086,5,1 M HCACO NCF PDO OUT ACK->Y:\$6C086,6,1 M HCACO NCF PDO IN CMD->Y:\$6C086,7,1 M HCACO NCF PD1 OUT ACK->Y:\$6C086,8,1 M HCACO NCF PD1 IN CMD->Y:\$6C086,9,1 M HCACO usHostFlags->X:\$6C086,0,16 M_HCAC0_HCF_HOST_COS_CMD->X:\$6C086,2,1 M HCAC0 HCF NETX COS ACK->X:\$6C086,3,1
M HCAC0 HCF SEND MBX CMD->X:\$6C086,4,1 M HCACO HCF RECV MBX ACK->X:\$6C086,5,1 M_HCAC0_HCF_PD0_OUT_CMD->X:\$6C086,6,1 M_HCAC0_HCF_PD0_IN_ACK->X:\$6C086,7,1 M_HCAC0_HCF_PD1_OUT_CMD->X:\$6C086,8,1 M_HCAC0_HCF_PD1_IN_ACK->X:\$6C086,9,1 M HCAC1_usNetxFlags->Y:\$6C087,0,16 M HCAC1_NCF_COMMUNICATING->Y:\$6C087,0,1 M HCAC1 NCF ERROR->Y:\$6C087,1,1 M_HCAC1_NCF_HOST_COS_ACK->Y:\$6C087,2,1 M HCAC1 NCF NETX COS CMD->Y:\$6C087,3,1 M HCAC1 NCF SEND MBX ACK->Y:\$6C087,4,1 M_HCAC1_NCF_RECV_MBX_CMD->Y:\$6C087,5,1 M HCAC1_NCF_PD0_OUT_ACK->Y:\$6C087,6,1
M_HCAC1_NCF_PD0_IN_CMD->Y:\$6C087,7,1 M HCAC1 NCF PD1 OUT ACK->Y:\$6C087,8,1 M_HCAC1_NCF_PD1_IN_CMD->Y:\$6C087,9,1 M HCAC1 usHostFlags->X:\$6C087,0,16 M HCAC1 HCF HOST COS CMD->X:\$6C087,2,1 M HCAC1 HCF NETX COS ACK->X:\$6C087,3,1 M_HCAC1_HCF_SEND_MBX_CMD->X:\$6C087,4,1
M_HCAC1_HCF_RECV_MBX_ACK->X:\$6C087,5,1 M HCAC1 HCF PD0 OUT CMD->X:\$6C087,6,1 M_HCAC1_HCF_PD0_IN_ACK->X:\$6C087,7,1 M_HCAC1_HCF_PD1_OUT_CMD->X:\$6C087,8,1

```
M_HCAC1_HCF_PD1_IN_ACK->X:$6C087,9,1
M CCO RCX APP COS APP READY->Y:$6C0C2,0,1
M_CCO_RCX_APP_COS_BUS_ON-YY:$6C0C2,1,1
M_CCO_RCX_APP_COS_BUS_ON_ENABLE->Y:$6C0C2,2,1
M_CCO_RCX_APP_COS_INIT->Y:$6C0C2,3,1
M CCO RCX APP COS INIT ENABLE->Y:$6C0C2,4,1
M_CC0_RCX_APP_COS_LOCK_CFG->Y:$6C0C2,5,1
M_CC0_RCX_APP_COS_LOCK_CFG_ENA->Y:$6C0C2,6,1
M CCO RCX APP COS DMA->Y:$6C0C2,7,1
M_CC0_RCX_APP_COS_DMA_ENABLE->Y:$6C0C2,8,1
M CC0 ulDeviceWatchdog->DP:$6C0C3
M CCO RCX COMM COS READY->Y:$6C0C4,0,1
M_CC0_RCX_COMM_COS_RUN->Y:$6C0C4,1,1
M_CC0_RCX_COMM_COS_BUS_ON->Y:$6C0C4,2,1
M_CC0_RCX_COMM_COS_CONFIG_LOCKED->Y:$6C0C4,3,1
M CCO RCX COMM COS CONFIG NEW->Y:$6C0C4,4,1
M_CC0_RCX_COMM_COS_RESTART_REQ->Y:$6C0C4,5,1
M_CCO_RCX_COMM_CO_REQ_ENA->Y:$6C0C4,6,1
M_CCO_RCX_COMM_COS_DMA->Y:$6C0C4,7,1
M CCO ulCommunicationState->DP:$6C0C5
M_CC0_ulCommunicationError->DP:$6C0C6
M CC0 usVersion->Y:$6C0C7,0,16
M CCO usWatchdogTime->X:$6C0C7,0,16
M_CC0_bPDInHskMode->Y:$6C0C8,0,8
M CC0 bPDInSource->Y:$6C0C8,8,8
M CC0 bPDOutHskMode->X:$6C0C8,0,8
M_CC0_bPDOutSource->X:$6C0C8,8,8
M_CC0_ulHostWatchdog->DP:$6C0C9
M CCO ulErrorCount->DP:$6C0CA
M CC0 bErrorLogInd->Y:$6C0CB,0,8
M_CC0_bErrorPDInCnt->Y:$6C0CB,8,8
M CC0 bErrorPDOutCnt->X:$6C0CB,0,8
M_CC0_bErrorSyncCnt->X:$6C0CB,8,8
M CC0 bSyncHskMode->Y:$6C0CC,0,8
M CC0 bSyncSource->Y:$6C0CC,8,8
M CCO ulSlaveState->DP:$6C0CE
M CCO ulSlaveErrLogInd->DP:$6C0CF
M_CC0_ulNumOfConfigSlaves->DP:$6C0D0
M CC0 ulNumOfActiveSlaves->DP:$6C0D1
M CC0 ulNumOfDiagSlaves->DP:$6C0D2
```

APPENDIX B – TURBO PMAC MEMORY MAPS

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
C-72EX Address	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX	\$6C000 netX
al-Port Memory Size	16384 bytes	8192 bytes	16384 bytes	8192 bytes	65536 bytes	8192 bytes	8192 bytes	16384 bytes	16384 bytes	65536 bytes	16384 bytes	16384 bytes	32768 bytes	32768 bytes
rice Number dware Assembly Options	1532410	1562420	1532510	1562520	1532500	1562540	1562740	1532100	1532100	1532100	1532100	1532100	1532100	1532100
Port 0	NOT CONNECTED	PROFIBUS	NOT CONNECTED	DEVICENET	NOT CONNECTED	CAN	CC-LINK	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)
Port 1 Port 2	NOT CONNECTED PROFIBUS	NOT AVAILABLE NOT AVAILABLE	NOT CONNECTED DEVICENET	NOT AVAILABLE NOT AVAILABLE	NOT CONNECTED CAN	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED
Port 3	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT CONNECTED	NOT AVAILABLE	NOT AVAILABLE	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED	NOT CONNECTED
cher Module License Information	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-	Unlimited number of master	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-	Unlimited number of master licenses	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-	Unlimited number of master	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-	Unlimited number of master licenses	Unlimited number of master licenses	(PROFIBUS Master) (CANopen Master) (DeviceNet Master) (AS-	Unlimited number of mast
	Interface Master) (PROFINET IO RT	licenses	Interface Master) (PROFINET IO RT	licenses	Interface Master) (PROFINET IO RT	licenses	licenses	Interface Master) (PROFINET IO RT	licenses	Interface Master) (PROFINET IO RT	licenses	licenses	Interface Master) (PROFINET IO RT	licenses
	Controller) (EtherCAT Master)		Controller) (EtherCAT Master)		Controller) (EtherCAT Master)			Controller) (EtherCAT Master)		Controller) (EtherCAT Master)			Controller) (EtherCAT Master)	
	(EtherNet/IP Scanner) (SERCOS III Master) 1 Master License		(EtherNet/IP Scanner) (SERCOS III Master) 1 Master License		(EtherNet/IP Scanner) (SERCOS III Master) 1 Master License			(EtherNet/IP Scanner) (SERCOS III Master) Unlimited number of		(EtherNet/IP Scanner) (SERCOS III Master) 1 Master License			(EtherNet/IP Scanner) (SERCOS III Master) 1 Master License	
								master licenses						
License Information rice Class	(SYCON.net) COMX 100	COMX 10	(SYCON.net) COMX 100	COMX 10	(SYCON.net) COMX 100	COMX 10	COMX 10	(SYCON.net) COMX 100	COMX 100	(SYCON.net) COMX 100	COMX 100	COMX 100	(SYCON.net) COMX 100	COMX 100
	COMX 100	COMX 10	COMX 100	COMATO	COMX 100	COMX 10	COMA 10	COMX 100	COMX 100	COMA 100	COMX 100	COMX 100	COMX 100	COMX 100
lock 0	System	Contant and	5	5 show	5	5	5 ml m	C. day	C	6	5 share	5	5	5
Channel Type Size of Channel	512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes
Channel Start Address	\$6CD00	\$6C000	\$6C000	\$6C000	\$6C000	\$6C000	\$6C000	\$6C000	\$6C000	\$6C000	\$6C000	\$6C000	\$6C000	\$6C000
Position of Handshake Cells netX System Flags Adress	IN HANDSHAKE CHANNEL X:\$6C080.0.8	IN HANDSHAKE CHANNEL X: \$60080.0.8	IN HANDSHAKE CHANNEL X:S6C080.0.8	IN HANDSHAKE CHANNEL X:\$6C080.0.8	IN HANDSHAKE CHANNEL X:\$6C080.0.8	IN HANDSHAKE CHANNEL X:56C080.0.8	IN HANDSHAKE CHANNEL X:\$6C080.0.8	IN HANDSHAKE CHANNEL X:\$6C080.0.8	IN HANDSHAKE CHANNEL X:\$6C080.0.8	IN HANDSHAKE CHANNEL X:S6C080.0.8	IN HANDSHAKE CHANNEL X:S6C080.0.8	IN HANDSHAKE CHANNEL X:S6C080.0.8	IN HANDSHAKE CHANNEL X:S6C080.0.8	IN HANDSHAKE CHANNEL X:S6C080.0.8
Host System Flags Adress	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8	X:\$6C080,8,8
Size of Handshake Cells	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS	8 BITS
Size of Mailbox Mailbox Start address	256 bytes \$6C040	256 bytes \$6C040	256 bytes \$6C040	256 bytes \$6C040	256 bytes \$60040	256 bytes \$6C040	256 bytes \$6C040	256 bytes \$6C040	256 bytes \$60040	256 bytes \$6C040	256 bytes \$6C040	256 bytes \$6CD40	256 bytes \$6C040	256 bytes \$6C040
Number of Subblocks	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Subblock 0	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Size	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes	176 bytes
Start Offset	\$6CD00	\$6C000	\$6C000	\$6C000	\$6000	\$6C000	\$6CD00	\$6C000	\$6CD00	\$6C000	\$6C000	\$6000	\$6000	\$6C000
Transfer Direction Transfer Type	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
- Subblock 1	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL
Size	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes	8 bytes
Start Offset Transfer Direction	\$6C02E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)	\$6002E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)	\$6002E OUT (Host System to netX)	\$6002E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)	\$6C02E OUT (Host System to netX)
Transfer Direction Transfer Type	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
- Subblock 2	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Size	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
Start Offset Transfer Direction	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)	\$6C030 IN (netX to Host System)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	U
- Subblock 3	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size Start Offset	128 bytes \$6C040	128 bytes \$60040	128 bytes \$6C040	128 bytes \$60040	128 bytes \$6C040	128 bytes \$6C040	128 bytes \$60040	128 bytes \$6C040	128 bytes \$6C040	128 bytes \$6C040	128 bytes \$60040	128 bytes \$6C040	128 bytes \$6C040	128 bytes \$60040
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)
Transfer Type	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode Handshake Bit	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED 4	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED 4	BUFFERED, HOST CONTROLLED 4	BUFFERED, HOST CONTROLLED 4	BUFFERED, HOST CONTROLLED 4	BUFFERED, HOST CONTROL 4
Subblock 4 Size	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes	MAILBOX 128 bytes
Start Offset	\$60060	\$60060	\$60060	\$60060	\$60060	\$6C060	\$60060	\$60060	\$60060	\$60060	\$60060	\$60060	\$60060	\$60060
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type Handshake Mode	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN
Handshake Bit	5	5	5	5	5	5	5	5	5	5	5	5	5	5
llock 1 Channel Type	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake
Size of Channel	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes
Channel Start Address	\$6C080	\$6C080	\$6C080	\$6C080	\$6C080	\$6C080	\$6C080	\$6C080	\$60080	\$6C080	\$6C080	\$6CD80	\$6C080	\$6C080
lock 2														
Channel Type Size of Channel	Communication	Communication 7424 bytes	Communication 15616 bytes	Communication 7424 but or	Communication 15616 bytes	Communication	Communication 7424 bytes	Communication	Communication 15616 bytes	Communication	Communication	Communication 15616 bytes	Communication	Communication 15616 bytes
Size of Channel Channel Start Address	15616 bytes \$6CDC0	7424 bytes \$6C0C0	15616 bytes \$6C0C0	7424 bytes \$6C0C0	15616 bytes \$6CDC0	7424 bytes \$6C0C0	\$6C0C0	15616 bytes \$6C0C0	15616 bytes \$6CDC0	15616 bytes \$6C0C0	15616 bytes \$6C0C0	15616 bytes \$6CDC0	15616 bytes \$6C0C0	15616 bytes \$6C0C0
Position of Handshake Cells	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL	IN HANDSHAKE CHANNEL
Size of Handshake Cells NetX Handshake Register	16 BITS Y:\$6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:S6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:S6C082.0.16	16 BITS Y:\$6C082.0.16	16 BITS Y:\$6C082.0.16
Netx Handshake Register Host Handshake Register	Y:\$6C082,0,16 X:\$6C082,0,16	X:\$6C082,0,16 X:\$6C082,0,16	X:\$6C082,0,16	¥:\$6C082,0,16 X:\$6C082,0,16	X:\$6C082,0,16 X:\$6C082,0,16	X:\$6C082,0,16	X:\$6C082,0,16	Y:\$6C082,0,16 X:\$6C082,0,16	X:\$6C082,0,16 X:\$6C082,0,16	X:\$6C082,0,16 X:\$6C082,0,16	X:\$6C082,0,16	X:\$6C082,0,16 X:\$6C082,0,16	X:\$6C082,0,16 X:\$6C082,0,16	X:\$6C082,0,16
Communication Class	MASTER	SLAVE	MASTER	SLAVE	MASTER	SLAVE	SLAVE	MASTER	SLAVE	SCANNER	ADAPTER	MESSAGING	IO-CONTROLLER	IO-DEVICE
Protocol Class Conformance Class	Managing Node	Managing Node	Server 0	Server 0	Scanner 0	Scanner 0	Adapter 0	Io-Controller	Io-Controller	lo-Device	lo-Device 0	Combination Firmware	Programmable Logic Controller (Plc)	Programmable Logic Cont 67
Number of Subblocks	9	9	9	9	9	9	9	9	9	9	9	9	9	9
	CONTE -	0017001	001700	001/7001	601/700	001/7001	CONTRO:	601/700 ⁻	CONTROL	50N700	001/7001	60NTD0/	0017001	00175 71
Subblock 0 Size	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes	CONTROL 8 bytes
Start Offset	\$6CDC2	\$6C0C2	\$6C0C2	\$6C0C2	\$60002	\$6002	\$6C0C2	\$6C0C2	\$6C0C2	\$6C0C2	\$6C0C2	\$6C0C2	\$6C0C2	\$6C0C2
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX
Transfer Type Handshake Mode	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	501 / 101 (T-T-T-	601 M (01) 671 7 17		COLUMN CTAT.			501 (1 (0)) (T) T) T	601 H 101 671	601 / 101 / CT17		00101001071717	00101001 CT1711		
Subblock 1 Size	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes	COMMON STATUS 64 bytes
Start Offset	\$6CDC4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4	\$6C0C4
Transfer Direction	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)
Transfer Type Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EXTENDED STATIS		EXTENDED CTATUS	EVERADED STATIS				EVTENDED STATUS	ENTENDED STATIS		EXTENDED STATUS		
Subblock 2 Size	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes	EXTENDED STATUS 432 bytes
Start Offset	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4	\$6C0D4
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type Handshake Mode	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		14411 DOV	1111001	111100	111100	MARDON	M48007	1111 DOI:	1441 POV	MAN DOV	MARDON	14100	1111007	
Subblock 3 Size	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes	MAILBOX 1600 bytes
Start Offset	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140	\$6C140
Transfer Direction	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX)	OUT (Host System to netX
	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Transfer Type Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTR

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
ubblock 4 size	MAILBOX 1600 bytes													
art Offset	\$6C2D0													
ansfer Direction ansfer Type	IN (netX to Host System) DPM (Dual-Port Memory)													
andshake Mode andshake Bit	UNKNOWN 5													
bblock 5	PROCESS DATA IMAGE													
ize tart Offset	5760 bytes \$6C4C0	1536 bytes \$60400	5760 bytes \$6C4C0	1536 bytes \$60400	5760 bytes \$6C4C0	1536 bytes \$60400	1536 bytes \$66400	5760 bytes \$6C4C0	5760 bytes \$66400	5760 bytes \$6C4C0	5760 bytes \$66400	5760 bytes \$66400	5760 bytes \$6C4C0	5760 bytes \$60400
Transfer Direction	OUT (Host System to netX)													
Transfer Type Handshake Mode	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROL												
Handshake Bit	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Subblock 6	PROCESS DATA IMAGE													
Size Start Offset	5760 bytes \$6CA60	1536 bytes \$6C640	5760 bytes \$6CA60	1536 bytes \$6C640	5760 bytes \$6CA60	1536 bytes \$6C640	1536 bytes \$6C640	5760 bytes \$6CA60						
Transfer Direction Transfer Type	IN (netX to Host System) DPM (Dual-Port Memory)													
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLL												
Handshake Bit	1	/	7	7	/	/	7	/	/	,	7	/	/	/
Subblock 7 Size	HIGH PRIORITY DATA IMAGE 64 bytes	HIGH PRIORITY DATA IMAGI 64 bytes												
Start Offset	\$6C460	\$6C460	\$6C460	\$6C460 OUT (Host System to netX)	\$6C460	\$6C460	\$6C460	\$6C460 OUT (Host System to netX)	\$6C460 OUT (Host System to netX)	\$6C460	\$6C460 OUT (Host System to netX)	\$6C460	\$6C460	\$6C460
Transfer Direction Transfer Type	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)
Handshake Mode Handshake Bit	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED 8	BUFFERED, HOST CONTROLI 8											
Subblock 8	HIGH PRIORITY DATA IMAGE													
lize	64 bytes													
itart Offset Transfer Direction	\$6C470 IN (netX to Host System)													
Fransfer Type Handshake Mode	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLI												
Handshake Mode Handshake Bit	9	9	9	9	9	9	9	9	9	9	9	9	9	9
ock 3														
hannel Type ize of Channel	Undefined 0 bytes	Communication 15616 bytes	Undefined 0 bytes	Undefined 0 bytes	Undefined 0 bytes	Communication 15616 bytes								
channel Start Address	\$6D000 BEGINNING OF CHANNEL	\$6C800 BEGINNING OF CHANNEL	\$6D000 BEGINNING OF CHANNEL	\$6C800 BEGINNING OF CHANNEL	\$6D000 BEGINNING OF CHANNEL	\$6C800 BEGINNING OF CHANNEL	\$6C800 BEGINNING OF CHANNEL	\$6D000 BEGINNING OF CHANNEL	\$6D000 BEGINNING OF CHANNEL	\$6D000 IN HANDSHAKE CHANNEL	\$6D000 BEGINNING OF CHANNEL	\$6D000 BEGINNING OF CHANNEL	\$6D000 BEGINNING OF CHANNEL	\$6D000
ize of Handshake Cells	NOT AVAILABLE	16 BITS	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	16 BITS								
etX Handshake Register ost Handshake Register	X:\$6D000 X:\$6D000,8,0	X:\$6C800 X:\$6C800,8,0	X:\$6D000 X:\$6D000,8,0	X:\$6C800 X:\$6C800,8,0	X:\$6D000 X:\$6D000,8,0	X:\$6C800 X:\$6C800,8,0	X:\$6C800 X:\$6C800,8,0	X:\$6D000 X:\$6D000,8,0	X:\$6D000 X:\$6D000,8,0	Y:\$6C083,0,16 X:\$6C083,0,16	X:\$6D000 X:\$6D000,8,0	X:\$6D000 X:\$6D000,8,0	X:\$6D000 X:\$6D000,8,0	Y:\$6C083,0,16 X:\$6C083,0,16
ommunication Class	UNDEFINED	MESSAGING	UNDEFINED	UNDEFINED	UNDEFINED	MESSAGING								
rotocol Class onformance Class	UNDEFINED 0													
umber of Subblocks	0	0	0	0	0	0	0	0	0	9	0	0	0	9
Subblock 0 Size										CONTROL 8 bytes				CONTROL 8 bytes
Start Offset										\$6D002				\$6D002
Transfer Direction Transfer Type										OUT (Host System to netX) DPM (Dual-Port Memory)				OUT (Host System to netX) DPM (Dual-Port Memory)
Handshake Mode Handshake Bit										UNCONTROLLED				UNCONTROLLED
										U				U
Subblock 1 Size										COMMON STATUS 64 bytes				COMMON STATUS 64 bytes
Start Offset Transfer Direction										\$6D004 IN (netX to Host System)				\$6D004 IN (netX to Host System)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode Handshake Bit										UNCONTROLLED 0				UNCONTROLLED 0
Subblock 2										EXTENDED STATUS				EXTENDED STATUS
Size										432 bytes				432 bytes
Start Offset Transfer Direction										\$6D014 IN (netX to Host System)				\$6D014 IN (netX to Host System)
Transfer Type Handshake Mode										DPM (Dual-Port Memory) UNCONTROLLED				DPM (Dual-Port Memory) UNCONTROLLED
Handshake Bit										0				0
Subblock 3										MAILBOX				MAILBOX
Size Start Offset										1600 bytes \$6D080				1600 bytes \$6D080
Transfer Direction										OUT (Host System to netX)				OUT (Host System to netX)
Transfer Type Handshake Mode										DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED				DPM (Dual-Port Memory) BUFFERED, HOST CONTROLI
Handshake Bit										4				4
Subblock 4										MAILBOX				MAILBOX
Size Start Offset										1600 bytes \$6D210				1600 bytes \$6D210
Transfer Direction Transfer Type										IN (netX to Host System) DPM (Dual-Port Memory)				IN (netX to Host System) DPM (Dual-Port Memory)
Handshake Mode Handshake Bit										UNKNOWN 5				UNKNOWN 5
Subblock 5 Size										PROCESS DATA IMAGE 5760 bytes				PROCESS DATA IMAGE 5760 bytes
Start Offset										\$6D400				\$6D400
Transfer Direction Transfer Type										OUT (Host System to netX) DPM (Dual-Port Memory)				OUT (Host System to netX) DPM (Dual-Port Memory)
Handshake Mode Handshake Bit										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROLL
										-				-
Subblock 6 Size										PROCESS DATA IMAGE 5760 bytes				PROCESS DATA IMAGE 5760 bytes
Start Offset Transfer Direction										\$6D9A0 IN (netX to Host System)				\$6D9A0 IN (netX to Host System)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode Handshake Bit										BUFFERED, HOST CONTROLLED 7				BUFFERED, HOST CONTROLL 7
ubblock 7										HIGH PRIORITY DATA IMAGE				HIGH PRIORITY DATA IMAGE
Size										64 bytes				64 bytes
Start Offset Transfer Direction										\$6D3A0 OUT (Host System to netX)				\$6D3A0 OUT (Host System to netX)
Transfer Type Handshake Mode										DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED				DPM (Dual-Port Memory) BUFFERED, HOST CONTROLL
Handshake Mode Handshake Bit										BUFFERED, HOST CONTROLLED 8				8
Subblock 8										HIGH PRIORITY DATA IMAGE				HIGH PRIORITY DATA IMAGI
Size										64 bytes				64 bytes
Start Offset Transfer Direction										\$6D3B0 IN (netX to Host System)				\$6D3B0 IN (netX to Host System)
Transfer Type Handshake Mode										DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED				DPM (Dual-Port Memory) BUFFERED, HOST CONTROLL
landshake Bit										9				9
ck 4 nannel Type	Undefined													

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP	EtherNet/IP	Open Modbus/TCP	PROFINET IO	PROFINET IO
										Scanner/Master	Adapter/Slave		Controller/Master	Device/Slave
Channel Start Address	\$6D000	\$6C800	\$6D000	\$6C800	\$6D000	\$6C800	\$6C800	\$6D000	\$6D000	\$6DF40	\$6D000	\$6D000	\$6D000	\$6DF40
osition of Handshake Cells	BEGINNING OF CHANNEL													
Size of Handshake Cells	NOT AVAILABLE													
NetX Handshake Register	X:\$6D000	X:\$6C800	X:\$6D000	X:\$6C800	X:\$6D000	X:\$6C800	X:\$6C800	X:\$6D000	X:\$6D000	X:\$6DF40	X:\$6D000	X:\$6D000	X:\$6D000	X:\$6DF40
Host Handshake Register	X:\$6D000,8,0	X:\$6C800,8,0	X:\$6D000,8,0	X:\$6C800,8,0	X:\$6D000,8,0	X:\$6C800,8,0	X:\$6C800,8,0	X:\$6D000,8,0	X:\$6D000,8,0	X:\$6DF40,8,0	X:\$6D000,8,0	X:\$6D000,8,0	X:\$6D000,8,0	X:\$6DF40,8,0
Communication Class	UNDEFINED													
Protocol Class	UNDEFINED													
Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lumber of Subblocks	0	0	0	0	0	0	0	0	0	0	0	0	0	0
llock 5														
Channel Type	Undefined													
Size of Channel	0 bytes													
Channel Start Address	\$6D000	\$6C800	\$6D000	\$60800	\$6D000	\$6C800	\$6C800	\$6D000	\$6D000	\$6DF40	\$6D000	\$6D000	\$6D000	\$6DF40
Position of Handshake Cells	BEGINNING OF CHANNEL													
Size of Handshake Cells	NOT AVAILABLE													
NetX Handshake Register	X:\$6D000	X:\$6C800	X:\$6D000	X:\$6C800	X:\$6D000	X:\$6C800	X:\$6C800	X:\$6D000	X:\$6D000	X:\$6DF40	X:\$6D000	X:\$6D000	X:\$6D000	X:\$6DF40
Host Handshake Register	X:\$6D000,8,0	X:\$6C800,8,0	X:\$6D000,8,0	X:\$6C800,8,0	X:\$6D000,8,0	X:\$6C800,8,0	X:\$6C800,8,0	X:\$6D000,8,0	X:\$6D000,8,0	X:\$6DF40,8,0	X:\$6D000,8,0	X:\$6D000,8,0	X:\$6D000,8,0	X:\$6DF40,8,0
Communication Class	UNDEFINED													
Protocol Class	UNDEFINED													
Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Number of Subblocks	0	0	0	0	0	0	0	0	0	0	0	0	0	0

APPENDIX C – POWER PMAC MEMORY MAPS

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
ACC-72EX Address netX Identification	Acc72EX[i].a	Acc72EX[i].a netX	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a	Acc72EX[i].a
Dual-Port Memory Size	16384 bytes	8192 bytes	16384 bytes	8192 bytes	65536 bytes	8192 bytes	8192 bytes	16384 bytes	16384 bytes	65536 bytes	16384 bytes	16384 bytes	32768 bytes	32768 bytes
Device Number Hardware Assembly Options	1532410	1562420	1532510	1562520	1532500	1562540	1562740	1532100	1532100	1532100	1532100	1532100	1532100	1532100
Port 0	NOT CONNECTED	PROFIBUS NOT AVAILABLE	NOT CONNECTED	DEVICENET NOT AVAILABLE	NOT CONNECTED	CAN NOT AVAILABLE	CC-LINK NOT AVAILABLE	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)	ETHERNET (internal Phy)
Port 1 Port 2	NOT CONNECTED PROFIBUS	NOT AVAILABLE	NOT CONNECTED DEVICENET	NOT AVAILABLE	NOT CONNECTED CAN	NOT AVAILABLE	NOT AVAILABLE	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED	ETHERNET (internal Phy) NOT CONNECTED
Port 3 Hilscher Module License Information	NOT CONNECTED (PROFIBUS Master) (CANopen	NOT AVAILABLE Unlimited number of master	NOT CONNECTED (PROFIBUS Master) (CANopen	NOT AVAILABLE Unlimited number of master	NOT CONNECTED (PROFIBUS Master) (CANopen	NOT AVAILABLE Unlimited number of master	NOT AVAILABLE Unlimited number of master	NOT CONNECTED (PROFIBUS Master) (CANopen	NOT CONNECTED Unlimited number of master	NOT CONNECTED (PROFIBUS Master) (CANopen	NOT CONNECTED Unlimited number of master	NOT CONNECTED Unlimited number of master	NOT CONNECTED (PROFIBUS Master) (CANopen	NOT CONNECTED Unlimited number of master
	Master) (DeviceNet Master) (AS-	licenses	Master) (DeviceNet Master) (AS-	licenses	Master) (DeviceNet Master) (AS-	licenses	licenses	Master) (DeviceNet Master) (AS-	licenses	Master) (DeviceNet Master) (AS-	licenses	licenses	Master) (DeviceNet Master) (AS-	licenses
	Interface Master) (PROFINET IO RT Controller) (EtherCAT Master)		Interface Master) (PROFINET IO RT Controller) (EtherCAT Master)		Interface Master) (PROFINET IO RT Controller) (EtherCAT Master)			Interface Master) (PROFINET IO RT Controller) (EtherCAT Master)		Interface Master) (PROFINET IO RT Controller) (EtherCAT Master)			Interface Master) (PROFINET IO RT Controller) (EtherCAT Master)	
	(EtherNet/IP Scanner) (SERCOS III		(EtherNet/IP Scanner) (SERCOS III		(EtherNet/IP Scanner) (SERCOS III			(EtherNet/IP Scanner) (SERCOS III		(EtherNet/IP Scanner) (SERCOS III			(EtherNet/IP Scanner) (SERCOS III	
	Master) 1 Master License		Master) 1 Master License		Master) 1 Master License			Master) Unlimited number of master licenses		Master) 1 Master License			Master) 1 Master License	
Tool License Information	(SYCON.net)		(SYCON.net)		(SYCON.net)			(SYCON.net)		(SYCON.net)			(SYCON.net)	
Device Class	COMX 100	COMX 10	COMX 100	COMX 10	COMX 100	COMX 10	COMX 10	COMX 100	COMX 100	COMX 100	COMX 100	COMX 100	COMX 100	COMX 100
Block 0					-	-	-	-			-			-
Channel Type Size of Channel	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes	System 512 bytes
Channel Start Address	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a	Acc72EX[i].Data8[0].a
Position of Handshake Cells netX System Flags Address	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a	IN HANDSHAKE CHANNEL Acc72EX[i].Data8[514].a
Host System Flags Address	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a 8 BITS	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a	Acc72EX[i].Data8[515].a 8 BITS	Acc72EX[i].Data8[515].a
Size of Handshake Cells Size of Mailbox	8 BITS 256 bytes	256 bytes	8 BITS 256 bytes	8 BITS 256 bytes	8 BITS 256 bytes	8 BITS 256 bytes	8 BITS 256 bytes	8 BITS 256 bytes	8 BITS 256 bytes	8 BITS 256 bytes	8 BITS 256 bytes	8 BITS 256 bytes	256 bytes	8 BITS 256 bytes
Mailbox Start address	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a	Acc72EX[i].Data8[256].a
Number of Subblocks	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Subblock 0	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Size Start Offset	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a	176 bytes Acc72EX[i].Data8[0].a
Transfer Direction	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional) DPM (Dual-Port Memory)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)	IN - OUT (Bi-Directional)
Transfer Type Handshake Mode	UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Subblock 1	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL	CONTROL
Size Start Offset	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a	8 bytes Acc72EX[i].Data8[184].a
Start Offset Transfer Direction	Acc72EX[I].Data8[184].a OUT (Host System to netX)	Acc/2EX[I].Data8[184].a OUT (Host System to netX)	Acc/2EX[I].Data8[184].a OUT (Host System to netX)	Acc/2EX[I].Data8[184].a OUT (Host System to netX)	OUT (Host System to netX)	Acc72EX[I].Data8[184].a OUT (Host System to netX)	Acc/2EX[I].Data8[184].a OUT (Host System to netX)	OUT (Host System to netX)	Acc/2EX[I].Data8[184].a OUT (Host System to netX)	Acc/2EX[I].Data8[184].a OUT (Host System to netX)	Acc/2EX[I].Data8[184].a OUT (Host System to netX)	Acc/2EX[I].Data8[184].a OUT (Host System to netX)	Acc72EX[i].Data8[184].a OUT (Host System to netX)	Acc72EX[I].Data8[184].a OUT (Host System to netX)
Transfer Type Handshake Mode	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory)	DPM (Dual-Port Memory) UNCONTROLLED	DPM (Dual-Port Memory) UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	COMMON STATUS		COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS		COMMON STATUS		COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS	COMMON STATUS
Subblock 2 Size	64 bytes	COMMON STATUS 64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	COMMON STATUS 64 bytes	64 bytes	COMMON STATUS 64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes
Start Offset Transfer Direction	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a IN (netX to Host System)	Acc72EX[i].Data8[192].a IN (netX to Host System)	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a IN (netX to Host System)	Acc72EX[i].Data8[192].a IN (netX to Host System)	Acc72EX[i].Data8[192].a IN (netX to Host System)	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a	Acc72EX[i].Data8[192].a IN (netX to Host System)
Transfer Type	IN (netX to Host System) DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	DPM (Dual-Port Memory)
Handshake Mode	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED	UNCONTROLLED
Handshake Bit	0	0	0	0	0	0	0	U	0	0	0	0	0	0
Subblock 3	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size Start Offset	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a	128 bytes Acc72EX[i].Data8[256].a
Transfer Direction	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)	OUT (Host System to netX) DPM (Dual-Port Memory)
Handshake Mode	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED
Handshake Bit	4	4	4	4	4	4	4	4	4	4	4	4	4	4
I Subblock 4	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size Start Offset	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX[i].Data8[384].a	128 bytes Acc72EX(i).Data8(384).a
Transfer Direction	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)	IN (netX to Host System)
Transfer Type Handshake Mode	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN	DPM (Dual-Port Memory) UNKNOWN
Handshake Bit	5	r.	-	5	5	5	5	5	5	5	5	5	5	5
	2	5	5	3										
- Block 1	, ,	5	5	,										
+ Block 1 Channel Type	Handshake	5 Handshake	5 Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake	Handshake
Channel Type Size of Channel	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes
Channel Type Size of Channel Channel Start Address	Handshake 256 bytes Acc72EX[i].Data8[512].a													
Channel Type	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes
Channel Type Size of Channel Channel Start Address Block 2 Channel Type Size of Channel	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes
Channel Type Size of Channel Channel Start Address Block 2 Channel Type Size of Channel Channel Start Address	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EV[i].Data8[512].a Communication 15616 bytes Acc72EV[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Ghannel Start Address Position of Handshake Cells Size of Handshake Cells	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc722K[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc72EX[i]Data8[512].a Communication 7424 bytes Acc72EX[i]Data8[768].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc722K[i].Data8[512].a Communication 15636 bytes Acc724C[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc722K(i).Data8[512].a Communication 7424 bytes Acc722K(i).Data8[768].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc722K[i].Data8[512].a Communication 15616 bytes Acc722K[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EV[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc72EK[i],Data8[512].a Communication 15616 bytes Acc72EV[],Data8[768].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc722K[i].Data8[512].a Communication 15616 bytes Acc722k[i].Data8[568].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc722K(i).Data8[512].a Communication 15636 bytes Acc722K(i).Data8[568].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc72EX(i).Data8[512].a Communication 15616 bytes Acc72EX(i).Data8[568].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc722K[i].Data8[512].a Communication 15616 bytes Acc722K[i].Data8[768].a IN HAND5HAKE CHANNEL 16 BITS	256 bytes Acc722K[i],Data8[512],a Communication 15616 bytes Acc722K[i],Data8[68],a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Size of Channel Start Address Position of Handshake Cells Size of Handshake Cells Size of Handshake Cells	256 bytes Acc72EX[i].Data8[512] a Communication 15616 bytes Acc72EX[i].Data8[768] a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520] a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a	256 bytes Acc72EX[I,Data8[512] a Communication 15616 bytes Acc72EX[I,Data8[56] a IN HANDENAKE CHANNEL 16 BITS Acc72EX[I],Data8[520] a	256 bytes Acc722K[i].Data8[512].a Communication 7424 bytes Acc722K[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc722K[i].Data8[520].a	256 bytes Acc722K[I].bata8[512].a Communication 15516 bytes Acc722K[I].bata8[576].a IN HANDENAKC CHANNEL 16 BITS Acc722K[I].bata8[520].a	255 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a IN HAND5HAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a	256 bytes Acc72EX[i],Data8[512] a Communication 15516 bytes Acc72EX[i],Data8[576] a IN HANDENAKE OHANNEL 16 BITS Acc72EX[i],Data8[520] a	256 bytes Acc722K[i],Data8[512],a Communication 15616 bytes Acc722K[i],Data8[568],a IN HANDSHAKE CHANNEL 16 BITS Acc722K[i],Data8[520],a	256 bytes Acc722K[i],Data8[512],a Communication 15636 bytes Acc722K[i],Data8[568],a IN HANDSHAKE CHANNEL 16 BITS Acc722K[i],Data8[520],a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a	256 bytes Acc72EX[i].Data8[512].a Communication 15516 bytes Acc72EX[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a	256 bytes Acc722EX[I,Data8[512].a Communication 15516 bytes Acc72EX[I,Data8[526].a IN HANDEAKE CHANNEL 16 BITS Acc722EX[I].Data8[520].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EV[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EV[i].Data8[520].a
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Position of Handshake Cells Size of Handshake Cells Next Handshake Register Host Handshake Register	256 bytes Acc72EX[i].Data8[512] a Communication 15616 bytes Acc72EX[i].Data8[768] a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520] a Acc72EX[i].Data8[522] a MASTER	256 bytes Acc722K[i].Data8[512].a Communication 7424 bytes Acc722K[i].Data8[768].a IN HANDSHAKC CHANNEL 16 BITS Acc728[i].Data8[520].a Acc722K[i].Data8[520].a Acc722K[i].Data8[522].a SLAVE	256 bytes Acc722K[i].Data8[512].a Communication 15636 bytes Acc724C[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS	256 bytes Acc7226/(j).Data8[512] a Communication 7424 bytes Acc7226/(j).Data8[768].a IN HANDSHAC CHANNEL 16 BITS Acc7226/(j).Data8[520].a Acc7226/(j).Data8[522].a SLAVE	256 bytes Acc722K[i].bata8[512].a Communication 15616 bytes Acc722K[i].bata8[576].a IN HANDSHAKE CHANNEL 16 BITS Acc722K[i].bata8[520].a Acc722K[i].bata8[522].a MASTER	256 bytes Acc72EX(I).Data8[512].a Communication 7424 bytes Acc72EX(I).Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX(I).Data8[520].a Acc72EX(I).Data8[522].a SLAVE	256 bytes Acc7225K[i].Data8[512].a Communication 7424 bytes Acc725K[i].Data8[768].a IN HANG5KAC GHANNEL 16 BITS Acc725K[i].Data8[520].a Acc725K[i].Data8[522].a SLAVE	256 bytes Acc722K(I,Data8[512] a Communication 15616 bytes Acc722K(I,Data8[520] a Acc722K(I,Data8[520] a Acc722K(I,Data8[520] a Acc722K(I,Data8[522] a MASTER	256 bytes Acc7226X[J].Data8[512].a Communication 15516 bytes Acc722K[J].Data8[580].a IN HANDSHAC CHANNEL 16 BITS Acc722K[J].Data8[520].a Acc722K[J].Data8[522].a SLAVE	256 bytes Acc7226(J].Data8[512].a Communication 15561 bytes Acc7226(J].Data8[58].a IN HANDSHAC CHANNEL 16 BITS Acc728(J].Data8[520].a Acc72720(J].Data8[520].a SCANNER	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[548].a IN HAND5HAKC CHANNEL 16 BITS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[522].a ADAPTER	256 bytes Acc72EX[i].Data8[512].a Communication 15516 bytes Acc72EX[i].Data8[540].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a	256 bytes Acc722EX[I,Data8[52].a Communication 15516 bytes Acc72EX[I,Data8[526].a IN HANDENAKC GHANNEL 16 BITS Acc722X[I].Data8[520].a Acc722X[I].Data8[522].a Acc722X[I].Data8[522].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a IN HANOSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[522].a IO-DEVICE
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Position of Handshake Cells Size of Handshake Cells Nett K Handshake Register Host Handshake Register Communication Class Protocol Class	256 bytes Acc72EX[i].Data8[512] a Communication 15616 bytes Acc72EX[i].Data8[768] a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520] a Acc72EX[i].Data8[520] a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a IN HANGSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[522].a	256 bytes Acc722K[J].Data8[512].a Communication 15636 bytes Acc722K[J].Data8[768].a N NANDSHAKE CHANNEL 16 BITS Acc722K[J].Data8[520].a Acc722K[J].Data8[522].a	256 bytes Acc722K[i].Data8[512].a Communication 7424 bytes Acc722K[i].Data8[768].a N HANDSHAKE CHANNEL 16 BITS Acc722K[i].Data8[520].a Acc722K[i].Data8[522].a	256 bytes Acc722EX[i].Data8[512].a Communication 15616 bytes Acc722EX[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc722EX[i].Data8[520].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[522].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a NHANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a N HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a	256 bytes Acc722K[i].Data8[512].a Communication 156316 bytes Acc722K[i].Data8[768].a Acc722K[i].Data8[520].a Acc722K[i].Data8[520].a	256 bytes Acc7220(J].0ata8[512].a Communication 156/16 bytes Acc7220(J].0ata8[512].a N HANDSHAKE CHANNEL 16 BITS Acc7220(J].0ata8[520].a Acc7220(J].0ata8[520].a	256 bytes Acc72EX[I].Data8[512].a Communication 15616 bytes Acc72EX[I].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a	256 bytes Acc722K[i,Data8[512].a Communication 15616 bytes Acc722K[i,Data8[768].a N HANISHAKE CHANNEL 16 BITS Acc722K[i,Data8[520].a Acc722K[i,Data8[520].a	256 bytes Acc722EX[i].Data8[512].a Communication 15616 bytes Acc722EX[i].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc722EX[i].Data8[520].a Acc722EX[i].Data8[520].a	256 bytes Acc722K[i].Data8[512].a Communication 15616 bytes Acc722K[i].Data8[768].a IN HANG5AKE CHANNEL 16 BITS Acc722K[i].Data8[520].a Acc722K[i].Data8[522].a Dio-DeVice
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Position of Handshake Cells Size of Handshake Cells Nett Knadshake Register Host Handshake Register	256 bytes Acc72EX[i].Data8[512] a Communication 15616 bytes Acc72EX[i].Data8[768] a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520] a Acc72EX[i].Data8[522] a MASTER	256 bytes Acc722K[i].Data8[512].a Communication 7424 bytes Acc722K[i].Data8[768].a IN HANDSHAKC CHANNEL 16 BITS Acc728[i].Data8[520].a Acc722K[i].Data8[520].a Acc722K[i].Data8[522].a SLAVE	256 bytes Acc722K[J].Data8[512].a Communication 15636 bytes Acc722K[J].Data8[768].a N NANDSHAKE CHANNEL 16 BITS Acc722K[J].Data8[520].a Acc722K[J].Data8[522].a	256 bytes Acc7226/(j).Data8[512] a Communication 7424 bytes Acc7226/(j).Data8[768].a IN HANDSHAC CHANNEL 16 BITS Acc7226/(j).Data8[520].a Acc7226/(j).Data8[522].a SLAVE	256 bytes Acc722K[i].bata8[512].a Communication 15616 bytes Acc722K[i].bata8[576].a IN HANDSHAKE CHANNEL 16 BITS Acc722K[i].bata8[520].a Acc722K[i].bata8[522].a MASTER	256 bytes Acc72EX(I).Data8[512].a Communication 7424 bytes Acc72EX(I).Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX(I).Data8[520].a Acc72EX(I).Data8[522].a SLAVE	256 bytes Acc7225K[i].Data8[512].a Communication 7424 bytes Acc725K[i].Data8[768].a IN HANG5KAC GHANNEL 16 BITS Acc725K[i].Data8[520].a Acc725K[i].Data8[522].a SLAVE	256 bytes Acc722K(I,Data8[512] a Communication 15616 bytes Acc722K(I,Data8[520] a Acc722K(I,Data8[520] a Acc722K(I,Data8[520] a Acc722K(I,Data8[522] a MASTER	256 bytes Acc7226X[J].Data8[512].a Communication 15516 bytes Acc722K[J].Data8[580].a IN HANDSHAC CHANNEL 16 BITS Acc722K[J].Data8[520].a Acc722K[J].Data8[522].a SLAVE	256 bytes Acc7226(J].Data8[512].a Communication 15561 bytes Acc7226(J].Data8[58].a IN HANDSHAC CHANNEL 16 BITS Acc728(J].Data8[520].a Acc72720(J].Data8[520].a SCANNER	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[548].a IN HAND5HAKC CHANNEL 16 BITS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[522].a ADAPTER	256 bytes Acc72EX[i].Data8[512].a Communication 15516 bytes Acc72EX[i].Data8[540].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a	256 bytes Acc722EX[I,Data8[52].a Communication 15516 bytes Acc72EX[I,Data8[526].a IN HANDENAKC GHANNEL 16 BITS Acc722X[I].Data8[520].a Acc722X[I].Data8[522].a Acc722X[I].Data8[522].a	256 bytes Acc725K(i) Data8[512].a Communication 15516 bytes Acc725K(i) Data8[768].a N HANDSHKE GHAWEL 15 BITS Acc722K(i) Data8[520].a Acc722K(i) Data8[520].a N-c725K(i) Data8[520].a N-c725K(i) Data8[520].a
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Ochannel Start Address Ochannel Start Address Position of Handhale Colls Position of Handhale Register Host Handshake Register Host Handshake Register Communication Class Protocol Class Conformance Class Number of Subblocks	256 bytes Acc72EX[I].Data8[512] a Communication 15516 bytes Acc72EX[I].Data8[56] a IN HANDGHARC (HANNEL 16 BITS Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[522] a MASTER Managing Node 0 9	256 bytes Acc722K[I].Data8[512].a Communication 7424 bytes Acc722K[I].Data8[768].a IN HANGHAKK CHANNEL 16 BITS Acc728K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a SLAVE Managing Node 0 9	256 bytes Acc722Cf[J].Data8[512].a Communication 15616 bytes Acc722Df[J].Data8[502].a IN HANGNEKC CHANNEL 16 BITS Acc722Df[J].Data8[520].a Acc722Df[J].Data8[520].a MASTER Server 0 9	256 bytes Acc722F(J],Data8[512] a Communication 7424 bytes Acc722F(J],Data8[768] a IN HANDSHAC CHANNEL 16 BITS Acc728F(J],Data8[520],a Acc722F(J],Data8[520],a SLAVE Server 0 9	256 bytes Acc722EX[I].Data8[512].a Communication 15616 bytes Acc72EX[I].Data8[576].a IN HANDENAKE CHANNEL 16 BITS Acc72EX[I].Data8[522].a MASTER Scanner 0 9	256 bytes Acc72EX[I].Data8[512].a Communication 7424 bytes Acc72EX[I].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a SLAVE Scanner 0 9	256 bytes Acc722K(i].Data8[512].a Communication 7424 bytes Acc722K(i].Data8[58].a IN HAROSHAKE CHANNEL I 6 BITS Acc722K(i].Data8[520].a Acc722K(i].Data8[522].a SLAVE 0 9	256 bytes Acr222K(I).Data8[512].a Communication 15616 bytes Acr222K(I).Data8[520].a IN HANGSHAK COMANNEL 16 BITS Acr222K(I).Data8[520].a Acr272K(I).Data8[520].a MASTER Io-Controller 0 9	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[56].a IN HANGHKAC CHANNEL 16 BITS Acc722K[I].Data8[520].a Acc722K[I].Data8[522].a SLAVE Io-Controller 0 9	256 bytes Acc722Cf(J),Data8[512],a Communication 15616 bytes Acc722Df(J),Data8[52],a Acc722Df(J),Data8[520],a Acc722Df(J),Data8[520],a Acc722Df(J),Data8[520],a SCANVER Io-Device 0 9	256 bytes Acc72EX[I].Data8[512].a Communication 15616 bytes Acc72EX[I].Data8[56].a IN HAND6HAKC CHANNEL 16 BITS Acc72EX[I].Data8[520].a Acc72EX[I].Data8[522].a ADAPTER b-Device 0 9	256 bytes Acc722K(j)Data8[512].a Communication 15616 bytes Acc722K(j)Data8[58].a IN HANGSHAC CHANNEL 16 BITS Acc722K(j)Data8[520].a Acc722K(j)Data8[520].a MESSA(N6 Combination Firmware 0 9	256 bytes Acc722EX[I,Data8[52],a Communication 15516 bytes Acc72EX[I,Data8[526],a IN HANDENAKE OHANNEL 16 BITS Acc72EX[I],Data8[522],a Acc72EX[I],Data8[522],a Acc72EX[I],Data8[522],a Dio-CONTROLER Programmable Logic Controller (Pic) 0 9	256 bytes Acc722K(i)Data8[512].a Communication 15616 bytes Acc722K(i)Data8[768].a Ni HANDSHAKE CHANNEL 16 BITS Acc722K(i)Data8[520].a Acc722K(i)Data8[522].a IO-DEVICE Programmable Logic Controller (67 9
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Channel Start Address Channel Start Address Channel Start Address Channel Start Address Size of Handshake Cells Size of Handshake Cells Communication Calss Protocol Class Comformance Class Number of Subblocks Size	256 bytes Acc72EX[I].Data8[512] a Communication 15516 bytes Acc72EX[I].Data8[56] a NH HANDGHAKE (HANNEL 16 BITS Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[522] a MASTER Managing Node 0 9 CONTROL 8 bytes	256 bytes Acc722K(I).Data8[512].a Communication 7424 bytes Acc722K(I).Data8[768].a IN HANGHAKK CHANNEL 16 BITS Acc728(I).Data8[520].a Acc722K(I).Data8[520].a Acc728(256 bytes Acc722Cf[J].Data8[512].a Communication 15616 bytes Acc722Cf[J].Data8[562].a IN HANGHAKC CHANNEL 16 BITS Acc722Cf[J].Data8[520].a Acc722Cf[J].Data8[520].a MASTER Server 0 9 CONTROL 8 bytes	256 bytes Acc722F(J],Data8[512],a Communication 7424 bytes IN HANGHARC CHANNEL 16 BITS Acc722F(J],Data8[520],a Acc722F(J],Data8[520],a Acc722F(J],Data8[520],a SLAVE Server 0 9 CONTROL 8 bytes	256 bytes Acc722RX[I,Data8[512].a Communication 15516 bytes Acc72EX[I,Data8[576].a IN HANDENAKE CHANNEL 16 BITS Acc72EX[I,Data8[522].a MASTER Scanner 0 9 CONTROL 8 bytes	256 bytes Acc72EX[I].Data8[512].a Communication 7424 bytes Acc72EX[I].Data8[768].a IN HAND5HAKC GHANNEL 16 BTS Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a StAVE Scanner 0 9 CONTROL 8 bytes	256 bytes Acc722K(I)Data8[512].a Communication 7424 bytes Acc722K(I)Data8[58].a IN HANGKAKE CHANNEL I 6 BITS Acc722K(I)Data8[520].a Acc722K(I)Data8[522].a SLAVE 0 9 CONTROL 8 bytes	256 bytes Acr225K[i_Data8[512] a Communication 1556 bytes Nr HANGSHAK CoHANNEL 16 BITS Acr225K[i]Data8[520] a Acr225K[i]Data8[520] a Acr2725K[i]Data8[522] a MASTER bc-Controller 0 9 CONTROL 8 bytes	256 bytes Acc722K[I]Data8[512].a Communication 15616 bytes Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a SLAVE Io-Controller 0 9 CONTROL 8 bytes	256 bytes Acc72eX[I].Data8[512].a Communication 15616 bytes Acc722X[I].Data8[52].a IN HANGHAC CHANNEL 16 BITS Acc722X[I].Data8[52].a Acc722X[I].Data8[52].a SCANNER Io-Device 0 9 CONTROL 8 bytes	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes M HAND6HAKC CHANNEL 16 BITS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a DADAPTER 0 9 CONTROL 8 bytes	256 bytes Acc722K(i)Data8[512].a Communication 15616 bytes Acc722K(i)Data8[580].a IN HANGSHAC CHANNEL 16 BITS Acc722K(i)Data8[520].a Acc722K(i)Data8[520].a Acc722K(i)Data8[520].a Acc722K(i)Data8[520].a Acc722K(i)Data8[520].a Acc722K(i)Data8[520].a Acc722K(i)Data8[520].a Acc722K(i)Data8[520].a Acc722K(i)Data8[52].a Acc722K(i)Data8[52].a Acc722K(i)Data8[52].a Acc722K(i)Data8[512].a Acc72K(i)Data8[52].a	255 bytes Acc722EX[I,Data8[52],a Communication 15516 bytes Acc72EX[I,Data8[52],a IN HANDENARC OHANNEL 16 BITS Acc72EX[I],Data8[52],a Acc72EX[I],Data8[52],a Acc72EX[I],Data8[52],a ID-CONTROLER Programmable Logic Controller (Pic) 0 9 CONTROL 8 bytes	256 bytes Acc728(I)Data8[512] a Communication 15616 bytes Acc728(I)Data8[768] a IN HANOSHAKE CHANNEL 16 BTS Acc728(I)Data8[520] a Acc7228(I)Data8[522] a IO-DEVICE Programmable Logic Controller 67 9 CONTROL 8 bytes
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Position of Handshake Cells Size of Handshake Cells Size of Handshake Cells Next Handshake Register Host Handshake Register Communication Class Protocol Class Conformance Class Number of Subblocks 	256 bytes Acc72EX[I],Data8[512] a Communication 15616 bytes Acc72EX[I],Data8[768] a NH ANDENARC GANANEL 16 BTS Acc72EX[I],Data8[520] a Acc72EX[I],Data8[520] a MASTER Maging Node 0 9 CONTROL 8 bytes Acc72EX[I],Data8[768] a	256 bytes Acc722K[i].Data8[512].a Communication 7424 bytes Acc722K[i].Data8[520].a Acc722K[i].Data8[520].a Acc722K[i].Data8[520].a Acc722K[i].Data8[520].a Acc72K[i].Data8[520].a Manging Node 9 CONTROL 8 bytes Acc72K[i].Data8[768].a	25b bytes Acc722K[J].Data8[512].a Communication 1556 bytes Acc722K[J].Data8[520].a Acc722K[J].Data8[520].a Acc722K[J].Data8[520].a Acc722K[J].Data8[520].a Acc722K[J].Data8[520].a MASTES CONTROL B bytes Acc72K[J].Data8[576].a	256 bytes Acc722K(J).Data8(512).a Communication 7424 bytes IN HANDSHARC CHANNEL 16 BITS Acc722K(J).Data8(520).a Acc722K(J).Data8(520).a Acc722K(J).Data8(522).a Sever 0 9 CONTROL 8 bytes Acc72K(J).Data8(768).a	256 bytes Acc7226X[i].bata8[512].a Communication 15516 bytes Acc722K[i].bata8[576].a IN HANDENAKC CHANNEL 16 BITS Acc7226X[i].bata8[52].a Acc7226X[i].bata8[52].a MASTEK Canner 0 9 CONTROL 8 bytes Acc722K[i].bata8[768].a	256 bytes Acc728X(I).Data8[512].a Communication 7424 bytes Acc728X(I).Data8[520].a IN HANDSHAKE CHANNEL 16 BTS Acc728X(I).Data8[520].a Acc728X(I).Data8[520].a Scamer 0 9 CONTROL 8 bytes Acc728X(I).Data8[768].a	256 bytes Acc22bK(I].Data8[512].a Communication 7424 bytes Acc722bK(I].Data8[768].a Ni HAKDEKK CHANNEL 16 BITS Acc722bK(I].Data8[520].a Acc722bK(I].Data8[520].a Acc722bK(I].Data8[520].a Akpter 0 9 CONTROL 8 bytes Acc722k(I].Data8[768].a	256 bytes Act22EX[i].bata8[512].a Communication 15616 bytes Nr HAUSEARC CHANNEL 16 BITS Act72EX[i].bata8[520].a Act72EX[i].bata8[520].a Act72EX[i].bata8[520].a MASTEN b Controller 0 9 9 CONTROL 8 bytes Act72EX[i].bata8[768].a	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes 15616 bytes 168175 Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[522].a is-Controller 0 9 CONTROL 8 bytes Acc72K[I].Data8[768].a	256 bytes Acc722K[I].Data8[512].a Communication 15561 bytes Acc722K[I].Data8[520].a Nr HANDSHAC CHANNEL 16 BITS Acc725K[I].Data8[520].a Acc725K[I].Data8[520].a Acc725K[I].Data8[520].a Solverke 0 9 CONTROL 8 bytes Acc725K[I].Data8[576].a	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[526].a IN HANDSHAK CHANNEL 16 8ITS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a CONTROL B bytes Acc72K[I].Data8[768].a	256 bytes Acc722K(I)Data8[512].a Communication 15616 bytes Acc722K(I)Data8[528].a IN HANDSHAC CHANNEL 16 BITS Acc722K(I)Data8[520].a Acc722K(I)Data8[522].a MCSV6(N)Data8[522].a MCSV6(N)Data8[52].a CONTROL 8 bytes Acc72K(I)Data8[768].a	256 bytes Acc722K[I],Data8[52],a Communication 15516 bytes Acc722K[I],Data8[520],a NH HANDENAKC CHANNEL 16 BITS Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a DCONTROL Bytes CONTROL Bytes Acc72K[I],Data8[768],a	256 bytes Acc7254(i).Data8[512].a Communication 15616 bytes Acc7254(i).Data8[768].a IN HANDSHAC CHANNEL 18 BIS Acc7254(i).Data8[520].a Acc7254(i).Data8[520].a Acc7254(i).Data8[520].a Control Logic Controller (9 CONTROL 8 bytes Acc7254(i).Data8[768].a
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Position of Handhake Cells Size of Handhake Cells Size of Handhake Cells NetX Handshake Register Host Handshake Register Communication Class Protocol Class Conformance Class Number of Subblocks 	256 bytes Acc72EX[I].Data8[512] a Communication 15616 bytes Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[522] a MARTER Managing Node 0 9 9 CONTROL 8 bytes Acc72EX[I].Data8[768] a OUT (Host System to netX) DPM (DusPort Menory)	256 bytes Acc722K[i].Data8[512].a Communication 7424 bytes Acc722K[i].Data8[520].a Acc722K[i].Data8[520].a Acc722K[i].Data8[520].a Acc722K[i].Data8[520].a Acc722K[i].Data8[520].a Acc72K[i].Data8[520].a Acc72K[i].Data8[520].a CONTROL 8 bytes Acc72K[i].Data8[768].a OUT (Host System to netX) DPM (Dua1-port Memory)	256 bytes Acc722K[J].Data8[512].a Communication 15616 bytes Acc722K[J].Data8[52].a IN HANDSHK CHANNEL 16 BITS Acc722K[J].Data8[520].a Acc722K[J].Data8[520].a Acc722K[J].Data8[520].a MASTER Server 0 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9	256 bytes Acc7225(J].Data8[512].a Communication 7424 bytes IN HANDSHARC CHANNEL 16 BITS Acc7252(J].Data8[520].a Acc7225(J].Data8[520].a Acc7225(J].Data8[520].a Suive Suive CONTROL 8 bytes Acc725(J].Data8[768].a OUT (Hoat System to netX) DPM (DuaH5676 Memory)	256 bytes Acc722K[i].bata8[512].a Communication 15516 bytes Acc72EK[i].bata8[576].a IN HANDENAKC CHANNEL 16 BITS Acc722K[i].bata8[520].a Acc722K[i].bata8[52].a MASTER Scanner 0 9 5 CONTROL 8 bytes Acc722K[i].bata8[768].a OUT (Host System to netX) OUT (Host System to netX)	256 bytes Acc728X(I),Data8[512],a Communication 7424 bytes Acc728X(I),Data8[520],a Acc728X(I),Data8[520],a Acc728X(I),Data8[520],a Acc728X(I),Data8[520],a Acc728X(I),Data8[520],a Acc728X(I),Data8[520],a Acc728X(I),Data8[520],a Scame 9 CONTROL 8 bytes Acc728X(I),Data8[768],a OUT (Host System to netX) DPM (DuaHPOT Memory)	256 bytes Acc72bX(I].Data8[512].a Communication 7424 bytes Acc72bX(I].Data8[548].a IN HANGENKE CHANNEL 16 BITS Acc72bX(I].Data8[520].a Acc72bX(I].Data8[520].a SLAVE 0 9 9 CONTROL 8 bytes Acc72bX(I].Data8[768].a OUT (Host System to netX) DPM (DuaHPort Memory)	256 bytes Ac:22EX[i].Data8[512].a Communication 156 IS bytes Ac:72EX[i].Data8[512].a Ni HAUGENKE CHANNEL 16 BITS Ac:72EX[i].Data8[520].a Ac:72EX[i].Data8[520].a MASTEN b Controller 0 9 9 CONTROL 8 bytes Ac:72EX[i].Data8[756].a OUT (Hext System to netX) DPM (DuaH-Port Memory)	256 bytes Acc722K[I].Data8[512].a Communication 15516 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[522].a SLWE CONTROL B bytes Acc722K[I].Data8[768].a OUT (Hoat System to netX) DPM (Dua8[768].a6 DPM (Dua8[768].a6 DPM (Dua8[768].a6 DPM (Dua8[768].a6 DPM (Dua9[768].a6 DPM (Dua9[768].a6 D	256 bytes Acc722 (K)[.Data8[512].a Communication 15616 bytes Acc722 (K)[.Data8[520].a Acc722 (K)[.Data8[520].a Acc722 (K)[.Data8[520].a Acc722 (K)[.Data8[520].a Acc72 (K)[.Data8[520].a CONTROL 8 bytes Acc728 (K)[.Data8[766].a OUT (Host System to netk) DPM (Dua147 for Memory)	256 bytes Acc722K(I).Data8(512).a Communication 15616 bytes Acc722K(I).Data8(526).a IN HANDSHAK CHANNEL 16 8ITS Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a CONTROL 8 bytes Acc72K(I).Data8(768).a OUT (Host System to netX) DPM (Dua1476rt Memory)	256 bytes Acc722K(I)Data8[512].a Communication 15616 bytes Acc722K(I)Data8[58].a IN HANDSKAC CHANNEL 16 BITS Acc722K(I)Data8[520].a Acc722K(I)Data8[522].a MIS3AG(ND) Comtinue to firm ware 0 9 CONTROL 8 bytes Acc722K(I)Data8[768].a CONTROL 9 bytes Acc722K(I)Data8[768].a OUT (Host System to netX) DPM (UnaHort Memory)	255 bytes Acc722K[I],Data8[52],a Communication 15516 bytes Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[522],a Do CONTROLE Programmable Logic Controller (PIc) 0 9 9 9 0 CONTROL 8 bytes Acc722K[I],Data8[768],a OUT (Hots System to netX) OUT (Hots System to netX)	256 bytes Acc725(i)Data8[512].a Communication 15616 bytes Acc725(i)Data8[768].a IN HANDSHAC CHANNEL 16 B15 Acc725(i)Data8[522].a Acc725(i)Data8[522].a Acc725(i)Data8[522].a CONTROL 8 bytes CONTROL 8 bytes Acc725(i)Data8[768].a OUT (Host System to netX) DPM (Dua1Port Memory)
Channel Type Sixe of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Channel Start Address Channel Start Address Contention of Handshake Gells Sixe of Handshake Cells Sixe of Handshake Cells Next Handshake Register Host Handshake Register Contention Class Contention Class Contention Class Contention Class Sixe Sixe Sixe Start Offset Transfer Urpetion	256 bytes Acc72EX[I].Data8[512] a Communication 15516 bytes Acc72EX[I].Data8[768] a NH HANDEAKE (FANAWEL 16 BITS Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[522] a MATSTR Managing Node 0 9 CONTROL 8 bytes Acc72EX[I].Data8[768] a OUT (Hots System to netX)	256 bytes Acc722K[i].Data8[512].a Communication 7424 bytes Acc722K[i].Data8[768].a IN HANDENAKC CHANNEL 16 BITS Acc728[i].Data8[520].a Acc722K[i].Data8[520].a Acc722K[i].Data8[520].a SLAVE 0 9 CONTROL 8 bytes bytes bytes hone 0 9 CONTROL 8 bytes bytes hone 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	256 bytes Acc722Cf[J].Data8[512].a Communication 1556 bytes Acc722K[J].Data8[58].a IN HANDSHAC CHANNEL 16 BITS Acc722K[J].Data8[520].a Acc722K[J].Data8[520].a MASTER Server 0 9 CONTROL 8 bytes 0 0 0 CONTROL 8 bytes 0 0 0 0 0 0 0 0 0 0 0 0 0	256 bytes Acc7225(ij.Data8[512].a Communication 7424 bytes Acc7225(ij.Data8[528].a IN HANSHAK CHANNEL 16 BITS Acc7252(ij.Data8[520].a Acc7225(ij.Data8[520].a SLAVE Server 0 9 CONTROL 8 bytes Acc7225(ij.Data8[528].a OUT (Hot5 System to netX)	256 bytes Acc722EX[i].Data8[512].a Communication 15516 bytes Acc72EX[i].Data8[576].a IN HANDENAKE CHANNEL 16 BITS Acc722X[i].Data8[522].a MASTER Scanner 0 9 CONTROL 8 bytes Acc722X[i].Data8[56].a ACc72EX[i].Data8[56].a ACC72EX[i].Data8[56].a ACC72EX[i].Data8[56].a ACC72EX[i].Data8[56].a ACC72EX[i].Data8[56].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a IN HANDSHAKC GHANNEL 16 BTIS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a SLAVE Scanner 0 9 CONTROL 8 bytes b tests Acc72EX[i].Data8[768].a OUT (HotS System to netX)	256 bytes Acc722K(i].Data8[512].a Communication 7424 bytes Acc722K(i].Data8[542].a IN HANGHAKE CHANNEL 16 BITS Acc722K(i].Data8[520].a Acc722K(i].Data8[522].a SLAVE 0 9 CONTROL 8 bytes Acc722K(i].Data8[528].a CONTROL 8 bytes Acc722K(i].Data8[54].a	256 bytes Acc722K(i].Data8[512].a Communication 15515 bytes Acc722K(i].Data8[520].a IN HANGSHAK CirkANNEL 16 BITS Acc722K(i].Data8[520].a Acc722K(i].Data8[522].a MASTER b.Controller 0 9 CONTROL 8 bytes Acc722K(i].Data8[768].a OUT (Hot 554m to netk)	256 bytes Acc722f(J)Data8[512].a Communication 15616 bytes Acc722f(J)Data8[56].a IN HANGHKAC CHANNEL 16 BITS Acc722f(J)Data8[520].a Acc722f(J)Data8[520].a SLAVE 0 9 CONTROL 8 bytes Acc722f(J)Data8[58].a OUT (Hot Styten to netX)	256 bytes Acc72eX[I].Data8[512].a Communication 1556 bytes Acc72eX[I].Data8[52].a IN HANDSHAC CHANNEL 16 BITS Acc72eX[I].Data8[52].a Acc72eX[I].Data8[52].a SCANNER Io-Device 0 9 CONTROL 8 bytes Acc72eX[I].Data8[76].a OUTROL 8 bytes	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes M HANDENAC CHANNEL 16 BITS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[522].a ADAPTER 0 9 CONTROL 8 bytes Acc722K[I].Data8[768].a OUT (Hots System to netX)	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[512].a IN HANGSHAC CHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a MESSAGN6 Combination Firmware 0 9 CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT (Hots System to netX)	255 bytes Acc722EX[I,Data8[52],a Communication 15516 bytes Acc72EX[I,Data8[520],a IN HANDEAKE CHANNEL 16 BITS Acc72EX[I,Data8[520],a Acc72EX[I,Data8[520],a Acc72EX[I,Data8[520],a Acc72EX[I,Data8[520],a DC-CONTROL Bytes Acc72EX[I,Data8[56],a Acc72EX[I,Data8[56],	256 bytes Acc72EX[i]Data8[512] a Communication 15616 bytes Acc72EX[i]Data8[768] a IN HANOSHAKE CHANNEL 16 BITS Acc72EX[i]Data8[520] a Acc72EX[i]Data8[520] a Acc72EX[i]Data8[522] a IO-DEVICE 9 rogrammable Logic Controller 67 9 CONTROL 8 bytes Acc72EX[i]Data8[768] a OUT (HotSYstem to netX)
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Channel Start Address Desition of Handshake Cells Size of Handshake Cells NetX Handshake Register Host Handshake Register Host Handshake Register Communication Class Protocol Class Number of Subblocks Size Start Offset Transfer Type Handshake Mode Handshake Bit	256 bytes Acc72EX[I].Data8[512] a Communication 15516 bytes Acc72EX[I].Data8[576] a Acc72EX[I].Data8[576] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a MATER Managing Node 0 9 CONTROL 8 bytes Acc72EX[I].Data8[576] a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0	256 bytes Acc722K(I).Data8[512].a Communication 7424 bytes IN HANDSHAKE CHANNEL 16 BITS Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a SLAVE 0 9 CONTROL 8 bytes Acc722K(I).Data8[768].a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0	256 bytes Acc722C(I), Data8[512] a Communication 15616 bytes Acc722C(I), Data8[502] a IN HANGNEKC CHANNEL 16 BITS Acc722C(I), Data8[520] a Acc722C(I), Data8[520] a Acc722C(I), Data8[520] a MASTER Server 0 9 CONTROL 8 bytes Acc722C(I), Data8[768] a OUT (Hots System to netX) DPM (Dual+Ort Memory) UNCONTROLLED 0	256 bytes Acc722F(I)[Data8[512].a Communication 7424 bytes Acc722F(I)[Data8[520].a Acc722F(I)[Data8[520].a Acc722F(I)[Data8[520].a Acc722F(I)[Data8[520].a SLVE Server 0 9 CONTROL 8 bytes Acc722F(I)[Data8[528].a OUT (Hot System to netX) DPM (Dual+Fort Menory) UNCONTROLLED 0	256 bytes Acc722EX[i].bata8[512].a Communication 15616 bytes Acc72EX[i].bata8[576].a IN HANDENAKE CHANNEL 16 BITS Acc72EX[i].bata8[522].a MASTER Acc72EX[i].bata8[522].a MASTER Scanner 0 9 CONTROL 8 bytes Acc72EX[i].bata8[576].a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0	256 bytes Acc72EX[I].Data8[512].a Communication 7424 bytes Acc72EX[I].Data8[768].a IN HANDSHAKE CHANNEL 16 BTS Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a SLAVE Scanner 0 9 CONTROL 8 bytes Acc72EX[I].Data8[768].a OUT (Hot35ysten to netX) DPM (Dual+Ort Memory) UNCONTROLLED 0	256 bytes Acc722K(I)Data8[512].a Communication 7424 bytes Acc722K(I)Data8[528].a IN HANDEMAC ENANNEL 16 BITS Acc722K(I)Data8[520].a Acc722K(I)Data8[520].a SLAVE 0 9 CONTROL 8 bytes Acc722K(I)Data8[528].a OUT (Hots System to netX) DPM (DataPort Memory) UNCONTROLLED 0	256 bytes Acr225K[i_Data8[512].a Communication 15616 bytes Acr272K(i].Data8[520].a NI HANGSHAK COMANNEL 16 BITS Acr272K(i].Data8[520].a Acr272K(i].Data8[520].a MASTER Io-Controller 0 9 CONTROL 8 bytes Acr272K(i].Data8[576].a OUT[HotsSystem to netX) DPM (Dual-Port Memory) UNCONTROLLED 0	256 bytes Acc722F(J] Data8[512] a Communication 15616 bytes Acc722F(J] Data8[50] a Acc722F(J] Data8[50] a Acc722F(J] Data8[50] a Acc722F(J] Data8[50] a Acc722F(J] Data8[50] a SLAVE Io-Controller 0 9 CONTROL 8 bytes Acc722F(J] Data8[76] a OUT (Hot System to netX) DPM (Data4Fort Memory) UNCONTROLLED 0	256 bytes Acc722Cf(J) Data8[512] a Communication 15616 bytes Acc722Cf(J) Data8[52] a IN HANDSHARC CHANNEL 16 BITS Acc722Cf(J) Data8[52] a Acc722Cf(J) Data8[52] a SCANNER Io-Device 0 9 CONTROL 8 bytes Acc722Cf(J) Data8[52] a SCANNER 0 9 CONTROL 8 bytes Acc722Cf(J) Data8[76] a OUT (Hot System to netX) DPM (Dual+Ort Memory) UMCONTROLLED 0	256 bytes Acc722X[I].Data8[512].a Communication 15616 bytes Acc722X[I].Data8[526].a NI HANDENACE CHANNEL 16 BITS Acc72X[I].Data8[520].a Acc722X[I].Data8[520].a Acc722X[I].Data8[522].a ADAPTER 0 Device 0 9 CONTROL 8 bytes Acc722X[I].Data8[768].a OUT (Hots System to netX) DPM (Dual+Ort Memory) UNCONTROLLED 0	256 bytes Acc72EX(i).Data8[512].a Communication 15616 bytes Acc72EX(i).Data8[580].a IN HANGSHAC CHANNEL 16 BITS Acc72EX(i).Data8[520].a Acc72EX(i).Data8[520].a Acc72EX(i).Data8[520].a MESSAGN6 Combination Firmware 0 9 CONTROL 8 bytes Acc72EX(i).Data8[768].a OUT (Hots System to netX) DPM (Data4Port Menory) UNCONTROLLED 0	256 bytes Acc722EX[I],Data8[512],a Communication 15516 bytes Acc72EX[I],Data8[520],a NIH ANDENAKE OHANNEL 16 BITS Acc72EX[I],Data8[522],a Acc72EX[I],Data8[522],a NCCONTROLER Programmable Logic Controller (Pic) 0 9 CONTROL 8 bytes Acc72EX[I],Data8[768],a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLED	256 bytes Acc725(i)Data8[512]a Communication 15516 bytes Acc7225(i)Data8[768]a N HANOSHAKE CHANNEL 16 BTS Acc7255(i)Data8[768]a Acc72525(i)Data8[522]a iO-DEVICE 67 9 CONTROL 8 bytes CONTROL 8 bytes DPM (DuaHPort Memory) UNCONTROLLED 0
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Position of Handshake Cells Size of Handshake Cells Size of Handshake Cells Size of Handshake Cells Next Handshake Register Host Handshake Register Communication Class Protocol Class Conformance Class Number of Subblocks Size Size Size Size Size Direction Transfer Direction Transfer Direction Transfer Direction Handshake Mode Handshake Mide	256 bytes Acc72EX[I].Data8[512].a Communication 15636 bytes Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Managing Node 0 9 CONTROL 8 bytes Acc72EX[I].Data8[768].a OUT (Host System to netX) DPM (Due1Fort Memory) UNCONTROLED 0 COMMON STATUS	256 bytes Acc728([i]Data8[512].a Communication 7424 bytes Mr HANDSHAC CHANNEL 16 BTS Acc728([i]Data8[520].a Acc728([i]Data8[520].a Acc728([i]Data8[520].a Acc728([i]Data8[520].a Acc728([i]Data8[520].a CONTROL 8 bytes Acc728([i]Data8[768].a OUT(Hots System to netX) DPM (Dua1+074 Memory) UNCONTROLLED 0 COMMON STATUS	256 bytes Acc7225(J],Data8[512],a Communication 15616 bytes Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7220(J],Data8[520],a OVTROL 8,bytes Acc720(J],Data8[768],a OVTROL 9 DPM (Dua4,Port Memory) UNCONTROLED 0 COMMON STATUS	256 bytes Acc7225(J].Data8[512].a Communication 7424 bytes Acc7225(J].Data8[528].a IN HANDSHKE CHANNEL 16 BITS Acc725(J].Data8[520].a Acc7225(J].Data8[520].a Acc7225(J].Data8[520].a Server 0 5 CONTROL 8 bytes Acc725(J].Data8[758].a OUT (Host System to netX) DPM (Dua14Port Memory) UNCONTROLLED 0 COMMON STATUS	255 bytes Acc7228(I).Data8(512).a Communication 15516 bytes Acc7228(I).Data8(576).a IN HANDSHAKE CHANNEL 16 BITS Acc7228(I).Data8(520).a Acc7228(I).Data8(52).a MASTER Scanner 0 9 9 CONTROL 8 bytes Acc7228(I).Data8(768).a OUT (Host System to netX) DPM (Dual-Port Memory) UNICONTROLLED 0 COMMON STATUS	256 bytes Acc72EX[I],Data8[512],a Communication 7424 bytes Acc72EX[I],Data8[768],a IN HANDSHAK CHANNEL 16 BITS Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a CONTROL 8 bytes Acc72EX[I],Data8[768],a OUT (Host System to netX) DPM (Dua1407 Memory) UNCONTROLLED 0 COMMON STATUS	256 bytes Acc728X(I].Data8[512].a Communication 7424 bytes Acc728X(I].Data8[58].a IN HANGENKE CHANNEL 16 BITS Acc728X(I].Data8[520].a Acc728X(I].Data8[520].a Acc728X(I].Data8[520].a SLAVE 0 9 9 9 9 9 9 0 0 0 0 0 0 0 0 0 0 0 0	256 bytes Ac:22EX(I).Data8[512].a Communication 156 Ib bytes 168 Ib 1945 [56].a IN HANGENKE GHANNEL 16 BITS Ac:72EX(I).Data8[520].a Ac:72EX(I).Data8[520].a Ac:72EX(I).Data8[520].a Ac:72EX(I).Data8[520].a MASTER b Controller 0 9 9 9 0 0 0 0 0 0 0 0 0 0 0 0 0	256 bytes Acc7225(J].Data8[512].a Communication 15516 bytes Acc7225(J].Data8[58].a IN HANDSHKK CHANNEL 16 BITS Acc725(J].Data8[52].a SLWE b-Controller 0 9 9 CONTROL 8 bytes Acc725(J].Data8[76].a OUT (Host System to netX) DPM (Dua140rt Memory) UNCONTROLLED 0 0 COMMON STATUS	256 bytes Acc722 Cf(I) Data8[512].a Communication 15616 bytes Acc722 Cf(I) Data8[520].a Acc722 Cf(I) Data8[520].a Acc722 Cf(I) Data8[520].a Acc722 Cf(I) Data8[520].a Acc722 Cf(I) Data8[520].a Acc722 Cf(I) Data8[520].a SCANNER b Obwice 0 9 9 CONTROL 8 bytes Acc722 (I) Data8[768].a OUT (Host System to netX) DPM (Dua14 Port Memory) UNCONTROLED 0 COMMON STATUS	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes M HANDSHAK CHANNEL 16 BITS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a CONTROL B bytes Acc722K[I].Data8[765].a OUT (Host System to netX) DPM (Dua14ort Memory) UNCONTROLLED 0 COMMON STATUS	256 bytes Acc722K(I).Data8[512].a Communication 15616 bytes Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a Acc722K(I).Data8[520].a Acc72K(I).Data8[520].a Combination Firmware 0 9 CONTROL 8 bytes Acc720K(I).Data8[768].a OUT (Host System to netX) DPM (DuaHort Memory) UNCONTROLED	255 bytes Acc722EX[I],Data8[522],a Communication 15516 bytes Acc722EX[I],Data8[520],a Acc722X[I],Data8[520],a Acc722X[I],Data8[520],a Acc722X[I],Data8[522],a Acc722X[I],Data8[522],a Hor-ContRoLLER Programmable Logic Controller (Pic) 0 9 9 9 9 9 9 9 9 9 9 9 9 9	256 bytes Acc725(i)Data8[512].a Communication 15616 bytes Acc725(i)Data8[768].a Ni HANDSAte CHANNEL 15 BiTS Acc725(i)Data8[520].a Acc7225(i)Data8[520].a Acc725(i)Data8[520].a Acc725(i)Data8[520].a CONTROL 8 bytes CONTROL 8 bytes Acc725(i)Data8[768].a OUT (Host System to netX) DPM (Oua1Port Memory) UNCONTROLLED 0 COMMON STATUS
Channel Type Sixe of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Position of Handshake Cells Sixe of Handshake Cells Next Handshake Cells Sixe of Handshake Cells Next Handshake Register Host Handshake Register Communication Class Protocol Class Conformance Class Number of Subblocks Sixe Sixe Start Offet Transfer Upicetion Transfer Upice Handshake Mode Handshake Mide Handshake Bit Sixe Sixe L	256 bytes Acc72EX[I].Data8[512].a Communication 15636 bytes Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[522].a Managing Node 0 9 CONTROL 8 bytes Acc72EX[I].Data8[768].a OUT [Host System to netX) DPM [Uba4]Fort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[I].Data8[784].a	256 bytes Acc722K[I].Data8[512].a Communication 7424 bytes Acc722K[I].Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc728(I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a OUTOOL 8 bytes Acc728(I].Data8[768].a OUT[Host System to netX]. DPM (Dub4Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc728(I].Data8[784].a	255 bytes Acc7225(J],Data8[512] a Communication 15616 bytes Acc7225(J],Data8[512] a IN HANDSHAC CHANNEL 16 BITS Acc728(J],Data8[520] a Acc7228(J],Data8[520] a Acc7228(J],Data8[520] a Acc728(J],Data8[520] a Acc728(J],Data8[520] a Bytes Acc728(J],Data8[768] a OUT (Host System to netk) DPM (Dua4 Fort Memory) UNCONTROLED D COMMON STATUS 64 bytes Acc728(J],Data8[784] a	256 bytes Acc7225(J],Data8[512],a Communication 7424 bytes Acc7225(J],Data8[52],a IN HANDSHKE CHANNEL 16 BITS Acc7252(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc725(J],Data8[520],a CONTROL 8 bytes Acc725(J],Data8[758],a OUT (Host System to netX) DPM (Dua140-rot Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc725(J],Data8[784],a	256 bytes Acc722K[I],Data8[52],a Communication 15616 bytes Acc722K[I],Data8[570],a Acc722K[I],Data8[52],a Acc722K[I],Data8[52],a Acc722K[I],Data8[52],a MASTER Scanner 0 9 9 9 0 5 bytes Acc722K[I],Data8[58],a 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0	256 bytes Acc72EX[I].Data8[512].a Communication 7424 bytes Acc72EX[I].Data8[520].a IN HANDSHAK CHANNEL 16 BITS Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a CONTROL 8 bytes Acc72EX[I].Data8[768].a OUT[Host System to netX]. DPM (Dub4Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc72EX[I].Data8[784].a	256 bytes Acc728X(I].Data8[512].a Communication 7424 bytes Acc728X(I].Data8[58].a IN HANGKAC ENANNEL 16 BITS Acc728X(I].Data8[520].a Acc728X(I].Data8[520].a Acc728X(I].Data8[522].a SLAVE 0 9 9 9 COMTROL 8.bytes Acc728X(I].Data8[528].a OUT [Host System to netX) DPM (DuaHort Memory) UNCONTROLED 0 0 COMMON STATUS 64 bytes Acc728X(I].Data8[784].a	256 bytes Ac:22EX(I).Data8[512].a Communication 156 Ib bytes Ac:72EX(I).Data8[512].a IN HANGENER CHANNEL 16 BITS Ac:72EX(I).Data8[52].a MASTER 0 Controller 0 9 9 COMTROL 8.bytes Ac:72EX(I).Data8[758].a OUT[Host System to netX). DPM (Dua14Port Memory) UNCONTROLLED 0 0 COMMON STATUS 64 bytes Ac:72EX(I).Data8[784].a	256 bytes Acc722K[I]Data8[512].a Communication 15616 bytes Acc722K[I]Data8[58].a IN HANDSHKE CHANNEL 16 BITS Acc722K[I]Data8[52].a SLAVE 0 0 9 9 CONTROL 8 bytes Acc72K[I]Data8[78].a OUT (Host System to netX) DPM (Dua14PM Memory) UNCONTROLLED 0 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a DPM (Dua14PM Memory) 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a DPM (Dua14PM Memory) 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a DPM (Dua14PM Memory) 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a DPM (Dua14PM Memory) 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a	256 bytes Acc7220(I)[Data8[512].a Communication 15616 bytes Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a SCANNE 0 0 9 9 0 CONTROL 8 bytes Acc720(I)[Data8[768].a OUT (Host System to netX) DPM (Dua1-674 Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc720(I)[Data8[764].a	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a CONTROL 9 bytes Acc722K[I].Data8[768].a OUT (Hoct System to netX) DPM (DuaHFort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a	256 bytes Acc722K(I).Data8(512).a Communication 15616 bytes Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Combination firmware 0 9 9 CONTROL 8 bytes Acc722K(I).Data8(768).a QUT (Host System to netX) DPM (DuaHort Memory). UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K(I).Data8(784).a	255 bytes Acc722EX[I],Data8[52],a Communication 15516 bytes Acc722X[I],Data8[520],a Acc722X[I],Data8[520],a Acc722X[I],Data8[520],a Acc722X[I],Data8[520],a Acc722X[I],Data8[52],a Ho-CONTROL Programmable Logic Controller (Pic) 0 9 9 CONTROL Subject Acc722X[I],Data8[76],a OUT (Host System to netX) DPM (DuaJ-bort Memory) UNCONTROLLED 0 0 COMMON STATUS 64 bytes Acc72X[I],Data8[784],a	256 bytes Acc7254(i).Data8[512].a Communication 15616 bytes Acc7254(i).Data8[768].a IN HANDSHAKE CHANNEL 16 BITS Acc7254(i).Data8[520].a Acc7254(i).Data8[520].a Acc7254(i).Data8[520].a Acc7254(i).Data8[520].a Acc7264(i).Data8[520].a Acc7264(i).Data8[768].a OUT (Host System to net N) DPM (Da14Per Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc7254(i).Data8[768].a
Channel Type Sixe of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Channel Start Address Contomic Start Address Foreiter Channel Start Address Sixe of Handshake Cells Sixe of Handshake Cells Next Kindshake Cells Sixe Sixe Sixe Sixe Sixe Sixe Sixe Sixe	256 bytes Acc72EX[I].Data8[512] a Communication 15616 bytes Acc72EX[I].Data8[576] a Acc72EX[I].Data8[576] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a Acc72EX[I].Data8[520] a MASTER Managing Node 0 9 CONTROL 8 bytes Acc72EX[I].Data8[576] a OUT (Hot4S System to metX) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[I].Data8[784] a Acc72EX[I].Data8[784] a Acc72EX[I].Data8[784] a NK (netX b Hot5 System)	256 bytes Acc722K[i].Data8[512].a Communication 7424 bytes Acc722K[i].Data8[768].a IN HANGHAKC CHANNEL 16 BITS Acc728[i].Data8[520].a Acc728[i].Data8[520].a Acc728[i].Data8[520].a Acc728[i].Data8[520].a Bits Acc728[i].Data8[768].a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc728[i].Data8[78].a M (netX to KS System)	256 bytes Acc722C(I)_Data8[512] a Communication 15616 bytes Acc722D(I)_Data8[522] a IN HANGHKAC CHANNEL 16 BITS Acc722D(I)_Data8[520] a Acc722D(I)_Data8[520] a Acc722D(I)_Data8[520] a MASTER Server 0 9 CONTROL 8 bytes Acc722D(I)_Data8[768] a OUT (Hot System to netX) DPM (Dual Fort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes	256 bytes Acc722F(J],Data8[512],a Communication 7424 bytes Acc722F(J],Data8[526],a IN HANDSHAC CHANNEL 16 BITS Acc722F(J],Data8[520],a Acc722F(J],Data8[520],a Acc722F(J],Data8[520],a SLAVE Server 0 9 CONTROL 8 bytes Acc722F(J],Data8[528],a OUT (Hot System to netX) DPM (Dual-Port Menory) UNCONTROLLED 0 COMMON STATUS 64 bytes IN (netX to dox System)	256 bytes Acc722EX[i].bata8[512].a Communication 15616 bytes Acc72EX[i].bata8[576].a IN HANDENAKE CHANNEL 16 BITS Acc72EX[i].bata8[522].a MASTER Acc72EX[i].bata8[522].a MASTER Scanner 0 9 CONTROL 8 bytes Acc72EX[i].bata8[56].a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[i].bata8[768].a Acc72EX[i].bata8[768].a Acc72EX[i].bata8[768].a Acc72EX[i].bata8[768].a Acc72EX[i].bata8[768].a IN (betk bots System)	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[768].a IN HAND5HAKC CHANNEL 16 BTS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a CONTROL 8 bytes CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT (Hot5 System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[768].a OUT (Hot5 System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes M (netX to Hot5 System)	256 bytes Acc722K(i].Data8[512].a Communication 7424 bytes Acc722K(i].Data8[52].a IN HANDSHAE CHANNEL 16 BITS Acc722K(i].Data8[520].a Acc722K(i].Data8[520].a Acc722K(i].Data8[522].a SLAVE 0 9 CONTROL 8 bytes Acc722K(i].Data8[528].a OUT (Hot Stytem to netX) DPM (Dual-Port Menony) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K(i].Data8[784].a IN (netX to hot Stytem)	256 bytes Acc722K(i].Data8[512].a Communication 15616 bytes Acc722K(i].Data8[520].a NI HANGSHAC CHANNEL 16 BITS Acc722K(i].Data8[520].a Acc722K(i].Data8[520].a MASTER io-Controller 0 9 CONTROL 8 bytes Acc722K(i].Data8[528].a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0 COMMON STATUS 66 bytes Acc722K(i].Data8[786].a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0	256 bytes Acc7225(J],Data8[512],a Communication 15616 bytes Acc7225(J],Data8[526],a IN HANGHKAC CHANNEL 16 BITS Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a SLAVE 0 9 CONTROL 8 bytes Acc7225(J],Data8[52],a SLAVE 0 0 CONTROL 8 bytes Acc7225(J],Data8[76],a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc7225(J],Data8[78],a IN (hetX to Kot System)	256 bytes Acc722E(J],Data8[512],a Communication 15616 bytes Acc722D(J),Data8[52],a Acc722D(J),Data8[52],a Acc722D(J),Data8[52],a Acc722D(J),Data8[52],a SCANVER Io-Device 0 9 CONTROL 8 bytes Acc722D(J),Data8[52],a SCANVER Device 0 9 CONTROL 8 bytes Acc722D(J),Data8[76],a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722D(J),Data8[78],a IN (netX to Ko System)	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[52].a ADAPTER b-Device 0 9 CONTROL 8 bytes Acc722K[I].Data8[768].a OUT (Hots System to netX) DPM (Dual+Ort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes M (netX to Kos System)	256 bytes Acc72EX(i).Data8[512].a Communication 15616 bytes Acc72EX(i).Data8[520].a Acc72EX(i).Data8[520].a Acc72EX(i).Data8[520].a Acc72EX(i).Data8[520].a Acc72EX(i).Data8[520].a MESSAGN6 Combination Firmware 0 9 CONTROL 8 bytes Acc72EX(i).Data8[58].a OUT (Hots System to netX) DPM (Data4Port Menony) UNCONTROLLED 0 COMMON STATUS 64 bytes	2 55 bytes Acc722EX[I,Data8[52],a Communication 1 5516 bytes Acc72EX[I,Data8[52],a MH ANOENAKC OFANNEL 16 BITS Acc72EX[I],Data8[52],a Acc72EX[I],Data8[52],a Acc72EX[I],Data8[52],a Ho-CONTROLER Programmable Logic Controller (Pic) 0 9 CONTROL 8 bytes Acc72EX[I,Data8[768],a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLED 0 0 COMMON STATUS 64 bytes	256 bytes Acc72EX[i]Data8[512]a Communication 15616 bytes Acc72EX[i]Data8[768]a NH ANDSAtex CHANNEL 16 BTS Acc72EX[i]Data8[520]a Acc72EX[i]Data8[520]a Acc72EX[i]Data8[520]a Acc72EX[i]Data8[520]a CONTROL 8 bytes Acc72EX[i]Data8[768]a OUT (HotSystem to netX) DPM (Dua4Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes N (netX to Hot System)
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Position of Handshake Cells Size of Handshake Cells Nett Handshake Cells Nett Handshake Register Host Handshake Register Communication Class Protocol Class Number of Subblocks Size Start Offset Transfer Direction Transfer Direction Transfer Direction Transfer Direction Size Start Offset Size Start Offset Transfer Direction Transfer Direction	256 bytes Acc72EX[I].Data8[512].a Communication 15636 bytes Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[522].a Managing Node 0 9 CONTROL 8 bytes Acc72EX[I].Data8[768].a OUT [Host System to netX) DPM [Uba4]Fort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[I].Data8[784].a	256 bytes Acc722K[I].Data8[512].a Communication 7424 bytes Acc722K[I].Data8[768].a IN HANDSHAKC CHANNEL 16 BITS Acc728(I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a OUTON. 8 bytes Acc728(I].Data8[768].a OUT[Host System to netX]. DPM (Dub4Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc728(I].Data8[784].a	256 bytes Acc722C(I)_Data8[512] a Communication 15616 bytes Acc722D(I)_Data8[502] a IN HANGHAK CHANNEL 16 BITS Acc722D(I)_Data8[520] a Acc722D(I)_Data8[520] a Acc722D(I)_Data8[520] a MASTER Server 0 9 CONTROL 8 bytes Acc722D(I)_Data8[768] a OUT (Hot System to netX) DPM (Dual Port Memory) UNCONTROLLED 0 COMMON STATUS 66 bytes Acc722D(I)_Data8[78] a UN (NetX Hot System)	256 bytes Acc7225(J],Data8[512],a Communication 7424 bytes Acc7225(J],Data8[52],a IN HANDSHKE CHANNEL 16 BITS Acc7252(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc725(J],Data8[520],a CONTROL 8 bytes Acc725(J],Data8[758],a OUT (Host System to netX) DPM (Dua140-rt Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc725(J],Data8[784],a	256 bytes Acc722K[I],Data8[52],a Communication 15616 bytes Acc722K[I],Data8[570],a Acc722K[I],Data8[52],a Acc722K[I],Data8[52],a Acc722K[I],Data8[52],a MASTER Scanner 0 9 9 9 0 5 bytes Acc722K[I],Data8[58],a 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0	256 bytes Acc72EX[I].Data8[512].a Communication 7424 bytes Acc72EX[I].Data8[520].a IN HANDSHAK CHANNEL 16 BITS Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a CONTROL 8 bytes Acc72EX[I].Data8[768].a OUT[Host System to netX]. DPM (Dub4Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc72EX[I].Data8[784].a	256 bytes Acc728X(I].Data8[512].a Communication 7424 bytes Acc728X(I].Data8[58].a IN HANGKAC ENANNEL 16 BITS Acc728X(I].Data8[520].a Acc728X(I].Data8[520].a Acc728X(I].Data8[522].a SLAVE 0 9 9 9 COMTROL 8.bytes Acc728X(I].Data8[528].a OUT [Host System to netX) DPM (DuaHort Memory) UNCONTROLED 0 0 COMMON STATUS 64 bytes Acc728X(I].Data8[784].a	256 bytes Ac:22EX(I).Data8[512].a Communication 156 Ib bytes Ac:72EX(I).Data8[512].a IN HANGENER CHANNEL 16 BITS Ac:72EX(I).Data8[52].a MASTER 0 Controller 0 9 9 COMTROL 8.bytes Ac:72EX(I).Data8[758].a OUT[Host System to netX). DPM (Dua14Port Memory) UNCONTROLLED 0 0 COMMON STATUS 64 bytes Ac:72EX(I).Data8[784].a	256 bytes Acc722K[I]Data8[512].a Communication 15616 bytes Acc722K[I]Data8[58].a IN HANDSHKE CHANNEL 16 BITS Acc722K[I]Data8[52].a SLAVE 0 0 9 9 CONTROL 8 bytes Acc72K[I]Data8[78].a OUT (Host System to netX) DPM (Dua14PM Memory) UNCONTROLLED 0 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a DPM (Dua14PM Memory) 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a DPM (Dua14PM Memory) 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a DPM (Dua14PM Memory) 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a DPM (Dua14PM Memory) 0 COMMON STATUS 64 bytes Acc72K[I]Data8[78].a	256 bytes Acc7220(I)[Data8[512].a Communication 15616 bytes Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a Acc7220(I)[Data8[520].a SCANNE 0 0 9 9 0 CONTROL 8 bytes Acc720(I)[Data8[768].a OUT (Host System to netX) DPM (Dua1-674 Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc720(I)[Data8[764].a	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a CONTROL 9 bytes Acc722K[I].Data8[768].a OUT (Hoct System to netX) DPM (DuaHFort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[512].a IN HANGSHAC CHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a MESSAGN6 Combination Firmware 0 9 CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT (Hots System to netX) DPM (Dual+Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[78].a IN (netX to hots System)	2 55 bytes Acc722EX[I].Data8[52].a Communication 1 5516 bytes Acc72EX[I].Data8[52].a MH ANDENAKC OHANNEL 16 BITS Acc72EX[I].Data8[52].a Acc72EX[I].Data8[52].a Acc72EX[I].Data8[52].a Ho-CONTROLER Programmable Logic Controller (Pic) 0 9 CONTROL 8 bytes Acc72EX[I].Data8[768].a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[I].Data8[768].a Acc72EX[I].D	256 bytes Acc725(i)Data8[512].a Communication 15616 bytes Acc725(i)Data8[768].a IN HANG94KE CHANNEL 16 BITS Acc725(i)Data8[520].a Acc7225(i)Data8[520].a Acc725(i)Data8[520].a Acc725(i)Data8[520].a Acc725(i)Data8[520].a Acc725(i)Data8[768].a OUT (Host Spatem to netX) DPM (Iola4Pert Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc725(i)Data8[768].a
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Size of Channel Size of Channel Address Size of Handdhale Colls Size of Handdhale Colls Number Of Subblocks Subblock O Size Start Offset Transfer Type Handhale Bloce Handhale Bloce Handhale Bloce Start Offset Size Start Offset Size Start Offset Transfer Direction Transfer Direction Size Start Offset Size Offset Handhale Bloce Handhale Bloce	256 bytes Acc72EX[i].Data8[512].a Communication 150516 bytes Acc72EX[i].Data8[768].a NH KANDSHAK GKANNEL 16 BITS Acc72EX[i].Data8[522].a MATER Managing Node 0 9 CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT [Host System to netX] DPM [Coal-Port Memory) UNCONTROLLED 0 COMMON STATUS 6 d bytes Acc72EX[i].Data8[764].a N(netX to Host System) DPM (Coal-Port Memory)	256 bytes Acc722K[I].Data8[512].a Communication 7424 bytes Acc722K[I].Data8[768].a IN HANDSHAKC CHANNEL 16 BITS Acc728K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a CONTROL 8 bytes Acc728K[I].Data8[768].a OUT (Hots System to netX) DPM (DuaHPort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc728K[I].Data8[784].a IN (netX to Hots System)	255 bytes Acc722C(J)[Data8[512] a Communication 1556 bytes Acc722C(J)[Data8[512] a IN HANDSHKC CHANNEL 16 BITS Acc722R(J][Data8[520] a Acc722R(J][Data8[520] a Acc722R(J][Data8[520] a Acc722R(J][Data8[520] a Acc722R(J][Data8[520] a Acc722R(J][Data8[520] a Bytes Acc722R(J][Data8[520] a COMMON STATUS 64 bytes Acc722R(J][Data8[784] a IN (netX to Mod System)) DPM (Dua16[784] a IN (netX to Mod System))	256 bytes Acc7225(J],Data8[512],a Communication 7424 bytes Acc7225(J],Data8[52],a IN HANDSHKE CHANNEL 16 BITS Acc7252(J],Data8[52],a Acc7225(J],Data8[52],a Acc7225(J],Data8[52],a SLVE G CONTROL 8 bytes Acc725(J],Data8[758],a Acc725(J],Data8[758],a Acc725(J],Data8[758],a Acc725(J],Data8[758],a Acc725(J],Data8[758],a OUT (Host System to netX) DPM (Dua9-Der Memory) UNCONTROLLED COMMON STATUS 64 bytes Acc725(J],Data8[784],a IN (netX to Jost System) DPM (Dua9-Der Memory)	256 bytes Acc722K[i].bata8[52].a Communication 15616 bytes Acc722K[i].bata8[576].a IN HANDSHAKE CHANNEL 16 BITS Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a MASTER Scanner 0 9 9 9 9 0 COMMON STATUS 64 bytes Acc722K[i].bata8[768].a OUT (Hox System to netX) DPM (Dual-Port Menony) UNCONTROLLED 0 0 COMMON STATUS 64 bytes Acc722K[i].bata8[764].a IN (netX to Hox System) DPM (Dual-Port Menony) DPM (Dual-Port Menony) DPM (Dual-Port Menony) DPM (Dual-Port Menony) DPM (Dual-Port Menony) DPM (Dual-Port Menony)	256 bytes Acc72EX[I],Data8[512],a Communication 7424 bytes Acc72EX[I],Data8[768],a IN HANDSHAK CHANNEL 16 BITS Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[768],a OUTFOL 8 bytes Acc72EX[I],Data8[768],a OUTFOL 9 bytes Acc72EX[I],Data8[768],a OUTFOL 10 bytes Acc72EX[I],Data8[784],a IN (neX to Host System) DPM (DuaH20TM Kenory)	256 bytes Acc722K(I].Data8[512].a Communication 7424 bytes Acc722K(I].Data8[512].a IN HANGKAC ENANNEL 16 BITS Acc722K(I].Data8[520].a Acc722K(I].Data8[520].a Acc722K(I].Data8[522].a SLAVE 0 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 0	256 bytes Ac:22EX[(].Data8[512].a Communication 156 Ib bytes Ac:72EX[(].Data8[52].a IN HANGEKE CHANNEL 16 BITS Ac:72EX[(].Data8[52].a MASTER b Controller 0 9 9 COMTROL 8 bytes Ac:72EX[(].Data8[763].a Ac:72EX[(].Data8[763].a Ac:72EX[(].Data8[763].a Ac:72EX[(].Data8[763].a COMMON STATUS 6 bytes Ac:72EX[(].Data8[764].a IN (netX to Hoat System) Det (Data9[764].a IN (netX to Hoat System) Det (Data9[764].a IN (netX to Hoat System)	256 bytes Acc722K[I]Data8[512].a Communication 15616 bytes Acc722K[I]Data8[582].a IN HANDSHKE CHANNEL 16 BITS Acc722K[I]Data8[522].a SLAVE 0 CONTROL 8 bytes Acc72K[I]Data8[52].a SLAVE 0 CONTROL 8 bytes Acc72K[I]Data8[78].a OUT (Host System to netX) DPM (Dua14PM Memory) UKCONTROLLED 0 COMMON STATUS 64 bytes Acc728(I],IData8[78].a IN (netXto host System) DPM (Dua14PM Memory) 10 COMMON STATUS 64 bytes Acc728(I],IData8[78].a IN (netXto host System) DPM (Dua14PM Memory)	256 bytes Acc722 (K)[.Data8[512].a Communication 15616 bytes Acc72 (K)[.Data8[512].a Acc72 (K)[.Data8[520].a Acc72 (K)[.Data8[520].a Acc72 (K)[.Data8[520].a Acc72 (K)[.Data8[520].a Acc72 (K)[.Data8[520].a Acc72 (K)[.Data8[520].a COMTROL Bytes Acc72 (K)[.Data8[520].a COMTROL COMTROL Device 0 COMTROL COMMON STATUS 64 bytes Acc72 (K)[.Data8[784].a IN (netX to host System) Dev (Dua1.67 K monoy) Dev (Dua1.67 K monoy)	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[520].a N HANDSHAK CHANNEL 16 BITS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a COMMON Starts b bytes Acc722K[I].Data8[768].a OUT (Hoct System to netX) D PM (Dua1Fort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a N (netX to Hoct System) D PM (Dua1Fort Memory)	256 bytes Acc722K(I).Data8(512).a Communication 15616 bytes Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Combination firmware 0 9 9 CONTROL 8 bytes Acc722K(I).Data8(768).a OUT (Host System to netX) DPM (DuaHort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K(I).Data8(784).a IN (netX to Host System) DPM (DuaHort Memory)	255 bytes Acc722K[I],Data8[52],a Communication 15516 bytes Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[522],a Acc722K[I],Data8[52],a Acc722K[I],Data8[52],a Acc722K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a Acc72K[I],Data8[76],a N(netX to Hot System) DPM (Dua)-Port Memory)	256 bytes Acc725(i)Data8[512].a Communication 15616 bytes Acc725(i)Data8[768].a IN HANG5445 CHANNEL 16 BITS Acc725(i)Data8[520].a Acc7226(i)Data8[520].a Acc7226(i)Data8[520].a Acc725(i)Data8[520].a Acc726(i)Data8[520].a Acc726(i)Data8[768].a OUT (Host System to netX) DPM (IoLaHort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc725(i)Data8[764].a IN (netX to Host System)
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Channel Start Address Channel Start Address Channel Start Address Channel Start Address Position of Handshale Register Host Handshake Register Host Handshake Register Des Handshake Register Communication Class Protocol Class Conformance Class Number of Subblocks Size Start Offset Transfer Direction Transfer Direction Start Offset Transfer Direction Start Offset Transfer Direction Transfer Direction Transfer Direction Transfer Direction Transfer Direction Transfer Direction Transfer Direction Transfer Direction Transfer Type Handshake Mode Handshake Mode Handshake Mode Handshake Mode Handshake Mode	256 bytes Acc72EX[i].Data8[512].a Communication 150516 bytes Acc72EX[i].Data8[576].a Ri HANDSHAKE (GANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a MATER Managing Node 0 9 COMTROL 8 bytes Ac72EX[i].Data8[768].a OUTICOL 9 COMMON STATUS 6 bytes Ac72EX[i].Data8[768].a OUTICOLED 0 COMMON STATUS 6 bytes Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a OUTICOLTROLLED 0 COMMON STATUS 6 bytes Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a	256 bytes Acc722K[I]Data8[512].a Communication 7424 bytes Acc722K[I]Data8[768].a IN HANDSHAK CHANNEL 16 BITS Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc72K[I]Data8[520].a COMTROL 8 bytes Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a COMMON STATUS G4 bytes Acc72K[I]Data8[784].a IN (net to Host System) DPM (Dua1671 Memory) UNCONTROLLED 0 EXTENDED STATUS	255 bytes Acc722C(I)[Data8[512] a Communication 1556 bytes Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[576] a Acc722C(I)[Data8[576] a Acc722C(I)[Data8[576] a Acc722C(I][Data8[576] a N(NCONTROLLED 0 COMMON STATUS 64 bytes Acc722C(I][Data8[576] a N(NCONTROLLED 0 COMMON STATUS	256 bytes Acc7225(J],Data8[512],a Communication 7424 bytes Acc7225(J],Data8[52],a IN HANDSHKE CHANNEL 16 BITS Acc7225(J],Data8[52],a Acc7225(J],Data8[52],a SLVE 0 9 9 CONTROL 8 bytes Acc7254(J],Data8[52],a SLVE 0 9 CONTROL 8 bytes Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a N, Inext to Host System) DPM (Dua1450rt Memory) UNCONTROLED 0 0 EXTENDED STATUS	256 bytes Acc722K[I].bata8[52].a Communication 15516 bytes Acc722K[I].bata8[52].a MATER IN HANDSHAKE CHANNEL 16 BITS Acc722K[I].bata8[52].a Acc722K[I].bata8[52].a MASTER 9 CONTROL 8 bytes Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a NCONTROLLED 0 COMMON STATUS 64 bytes Acc72K[I].bata8[578].a NCONTROLLED 0 EXTENDED STATUS	256 bytes Acc72EX[I].Data8[512].a Communication 7424 bytes Acc72EX[I].Data8[768].a IN HANDSHAK CHANNEL 16 BTS Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a SchWE Scanner 0 9 CONTROL 8 bytes Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[764].a N(neX to Host System) DPM (Dua167 Memory) UNCONTROLLED 0 EXTENDED STATUS	256 bytes Acc725K(I].Data8[512].a Communication 7424 bytes Acc725K(I].Data8[512].a IN HANGKAC GHANNEL 16 BITS Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a Acc725K(I].Data8[522].a SLAVE 0 9 9 CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 CONTROL 8 bytes CONTROL 9 CONTROL	256 bytes Ac:22EX[(].Data8[512].a Communication 156 Ib bytes Ac:72EX[(].Data8[52].a IN HANGSKRC CHANNEL 16 BITS Ac:72EX[(].Data8[52].a MASTER IN CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 CONTROL 9	256 bytes Acc722K[I]Data8[512].a Communication 15616 bytes Acc722K[I]Data8[52].a XC272K[I]Data8[52].a Acc722K[I]Data8[52].a SLWE 0 0 9 CONTROL 8 bytes Acc72K[I]Data8[76].a Acc72K[I]Data8[76].a Acc72K[I]Data8[76].a Acc72K[I]Data8[76].a Acc72K[I]Data8[76].a Acc72K[I]Data8[76].a Acc72K[I]Data8[76].a COMMON STATUS 64 bytes Acc72K[I]Data8[76].a N(netXto1635 yetem) DPM (Data8[76].a Acc72K[I]Data8[76].a N(netXto1635 yetem) DPM (Data8[76].a N(netXto1635 yetem) DTM (Data8[76].a N(netXto1635 yetem)	256 bytes Acc722C(J) Data8[512].a Communication 15616 bytes Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a SCANNEC 0 9 9 CONTROL 8 bytes Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[576].a O COMMON STATUS 64 bytes Acc722C(J) Data8[576].a N(netX to Hood System) DPM (Data767 Memory) UNCONTROLLED 0 EXTENDED STATUS	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc72K[I].Data8[520].a Acc72K[I].Data8[520].a COMTROL B bytes Acc72K[I].Data8[76].a Acc72K[I].Data8[76].a COMMON STATUS 64 bytes Acc72K[I].Data8[764].a N (neX16 Host System) DPM (Data8[764].a N (neX16 Host System) DPM (Data6[764].a N (neX16 Host System) DTM (Data6[764].a N (neX16 Host System) DT	256 bytes Acc722K(I)Data8(512).a Communication 15616 bytes Acc722K(I)Data8(512).a IN HANGKAC CHANNEL 16 BITS Acc722K(I)Data8(520).a Acc722K(I)Data8(520).a Acc722K(I)Data8(520).a Acc722K(I)Data8(520).a Acc722K(I)Data8(520).a MESSAGING Combination Firmware 0 9 CONTROL 8 bytes Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a N(MACONTROLLED 0 COMMON STATUS 66 bytes Acc72K(I)Data8(784).a IN(neX to Host system) DPM (DatA9CT Memory) UNCONTROLED 0 EXTENDED STATUS	255 bytes Acc722K[I],Data8[512].a Communication 15516 bytes Acc722K[I],Data8[520].a Acc722K[I],Data8[520].a Acc722K[I],Data8[22].a Ino-CONTROLES Programmable Logic Controller (Pic) 0 9 CONTROL 8 bytes Acc722K[I],Data8[76].a Acc722K[I],Data8[76].a Acc722K[I],Data8[76].a Acc722K[I],Data8[76].a Acc722K[I],Data8[76].a Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Downorthe Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Downorthe Acc72K[I],Data8[76].a INCONTROLED 0 COMMON STATUS 64 bytes Acc72K[I],Data8[76].a IN(netX to Host System) DPM (Dual-Port Memory) UNCONTROLED 0 EXTENDED STATUS	256 bytes Acc72EV(i)Data8[512] a Communication 15616 bytes Acc72EV(i)Data8[768] a IN HANDSHAC CHANNEL 16 BITS Acc72EV(i)Data8[520] a Acc72EV(i)Data8[520] a Acc72EV(i)Data8[520] a Acc72EV(i)Data8[520] a Acc72EV(i)Data8[520] a Acc72EV(i)Data8[520] a Acc72EV(i)Data8[520] a COMTROL 8 bytes Acc72EV(i)Data8[768] a COMTROL 8 bytes Acc72EV(i)Data8[768] a COMTROL 8 bytes Acc72EV(i)Data8[768] a COMTROL 8 bytes Acc72EV(i)Data8[768] a COMTROL 8 bytes Acc72EV(i)Data8[768] a COMTROL 8 bytes Acc72EV(i)Data8[768] a COMTROL Bottes 0 COMTROL Bottes 0 EXTENDED STATUS
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Ochannel Start Address Dexition of Handshake Cells Size of Handshake Cells Size of Handshake Cells NetX Handshake Register Host Handshake Register Communication Class Protocol Class Conformance Class Conformance Class Conformance Class Conformance Class Conformatic Cells Size Start Offreit Handshake Bit Size Start Offreit Size Start Offreit Size Start Offreit Size Start Offreit Size Start Offreit Transfer Direction Transfer Direction Transfer Direction Transfer Type Handshake Bit Size Start Offreit Transfer Direction Transfer Direction Transfer Direction Transfer Direction Transfer Direction Transfer Street Size Size Size Start Offreit Transfer Direction	256 bytes Acc72EX[I].Data8[512].a Communication 15616 bytes Acc72EX[I].Data8[50].a Acc72EX[I].Data8[50].a Acc72EX[I].Data8[50].a Acc72EX[I].Data8[50].a Acc72EX[I].Data8[50].a Montes Managing Node 0 9 CONTROL 8 bytes CONTROL 8 bytes COMMON STATUS 6 d bytes COMMON STATUS 6 d bytes	256 bytes Acc722K[I].Data8[512].a Communication 7424 bytes Acc722K[I].Data8[520].a MANDARC CHANNEL 16 BTS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Managing Node 0 9 CONTROL 8 bytes Acc72K[I].Data8[768].a OUT (Host System to netX) DPM (Dua1-Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc72K[I.Data8[5768].a Mirkes Mir	256 bytes Acc7220(I)[Data8[512]] Communication 15616 bytes Acc7220(I)[Data8[520]] NI HANDSHAE CHANNEL 16 BTS Acc7220(I][Data8[520]] Acc7220(I][Data8[520]] Acc7220(I][Data8[520]] MATTE Server 0 9 CONTROL 8 bytes Acc7220(I][Data8[520]] CONTROL 8 bytes Acc7220(I][Data8[520]] CONTROL 9 CONTROL 8 bytes Acc7220(I][Data8[520]] CONTROL 8 bytes Acc7220(I][Data8[5768]] Acc7220(I][Data8[5768]] Acc7220(I][Data8[5768]] Acc7220(I][Data8[5768]] Acc7220(I][Data8[5768]] Acc7220(I][Data8[5768]] Acc7220(I][Data8[5768]] Acc7220(I][Data8[5768]] CONTROLED 0 COMMON STATUS 56 bytes Acc7220(I][Data8[5768]] Acc7200(I][Data8[5768]] Acc7200(I][Data8[256 bytes Acc7225(J],Data8[512],a Communication 7424 bytes N HANDSHARC CHANNEL 16 BTS Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[522],a Sever 0 9 CONTROL 8 bytes Acc725(J],Data8[768],a OUT (Host System to netX) DPM (Dua14Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes N17262(J) bata8[768],a N17262(J) bata8[768],a COMMON STATUS 64 bytes N17262(J) bata8[768],a N17262(J) bata8[768],a N17262(J) bata8[768],a N17262(J) bata8[768],a DPM (Dua14Port Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes	256 bytes Acc722K[i].bata8[512].a Communication 15516 bytes Acc722K[i].bata8[576].a IN HANDSHAKE CHANNEL 16 BITS Acc722K[i].bata8[520].a Acc722K[i].bata8[52].a MASTER Acc722K[i].bata8[52].a MASTER CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes COMMON STATUS 64 bytes Acc72K[i].bata8[768].a OUT (Host System to netX) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72K[i].bata8[768].a Acc72K[i].bata8[768].a Acc72K[i].bata8[768].a COMMON STATUS 64 bytes Acc72K[i].bata8[768].a Acc72K[i].bata8[768].a Acc72K[i].bata8[768].a Acc72K[i].bata8[768].a COMMON STATUS 64 bytes COMMON STATUS 64 bytes	256 bytes Acc72EV(I),Data8[512],a Communication 7424 bytes Acc72EV(I),Data8[768],a IN HANDSHAE CHANNEL 16 BTS Acc72EV(I),Data8[520],a Acc72EV(I),Data8[520],a Acc72EV(I),Data8[520],a Acc72EV(I),Data8[520],a Acc72EV(I),Data8[768],a CONTROL 8 bytes Acc72EV(I),Data8[768],a Acc72EV(I),	256 bytes Acc228([].Data8[512].a Communication 7424 bytes Acc728([].Data8[512].a NI HAUGHKE CHANNEL 16 BITS Acc728([].Data8[520].a Acc728([].Data8[520].a Acc728([].Data8[520].a Acc728([].Data8[520].a Acc728([].Data8[520].a Acc728([].Data8[520].a Acc728([].Data8[520].a Acc728([].Data8[520].a CONTROL 8 bytes Acc728([].Data8[520].a OUT (Host System to netX) DPM (Dua14Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc728([].Data8[548].a Acc728([].Data8[548].a CONTROLLED 0 CONTROLLED 0 EXTENDED STATUS 42 bytes	256 bytes Ac:22EX[I].Data8[512].a Communication 156 IS bytes Ac:72EX[I].Data8[512].a Nr HAUSEARC CHANNEL 16 BITS Ac:72EX[I].Data8[520].a Ac:72EX[I].Data8[522].a MGSTR Data8[52].a MGSTR Data8[52].a CONTROL 8 bytes Ac:72EX[I].Data8[768].a OUT (Host System to netX) DPM (Dua14Ort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Ac:72EX[I].Data8[768].a Ac:72EX[I].Data8	256 bytes Acc722K[I]Data8[512].a Communication 15616 bytes Acc722K[I]Data8[52].a IN HANDSHAC CHANNEL 16 BITS Acc722K[I]Data8[520].a Acc722K[I]Data8[522].a Ib Acc722K[I]Data8[522].a Ib Acc722K[I]Data8[522].a Ib Acc722K[I]Data8[52].a Ib Acc722K[I]Data8[52].a Decommon 9 CONTROL 8 bytes Acc722K[I]Data8[76].a OUT (Host System to netX) DPM (Dua14Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Att 2004 Data8[748].a N1772KID Data8[748].a	256 bytes Acc722 C(I)[Data8[512].a Communication 15616 bytes Acc722 C(I)[Data8[520].a Acc722 C(I)[Data8[520].a Acc723 C(I)[Data8[520].a Acc723 C(I)[Data8[520].a Acc723 C(I)[Data8[520].a Bytes CONTROL Bytes Acc728 C(I)[Data8[520].a CONTROL Bytes Acc728 C(I)[Data8[576].a OUT (Host System to netX) DPM (Dua1-674 Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acr628 C(I) Aba8[576].a Def (Dua1-674 Memory) UNCONTROLLED 0 EXTRUDED STATUS 432 bytes	256 bytes Acc722K(I).Data8(512).a Communication 15616 bytes Acc722K(I).Data8(526).a N HANDSHAK CHANNEL 16 BITS Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a Acc722K(I).Data8(520).a CONTROL B bytes Acc72K(I).Data8(768).a OUT (HotS System to netX) DPM (Dua1476rt Memory) UNCONTROLED 0 COMMON STATUS 64 Dytes Acc72K(I).Data8(768).a OUT (HotS System to netX) DPM (Dua1476rt Memory) UNCONTROLED 0 CONTROLED 0 CONTROLED 0 EXTENDED STATUS 432 bytes	256 bytes Acc722K(I)Data8(512)a Communication 15616 bytes Acc722K(I)Data8(52)a Acc722K(I)Data8(52)a Acc722K(I)Data8(52)a Acc722K(I)Data8(52)a Acc722K(I)Data8(52)a Acc722K(I)Data8(52)a Combination Rimware Combination Rimware Combination Rimware Comtrol Bytes Acc722K(I)Data8(76)a Acc72K(I)Data8(76)a CONTROL Bytes Acc72K(I)Data8(76)a CONTROL Bytes Acc72K(I)Data8(76)a COMMON STATUS 64 bytes Acc72K(I)Data8(76)a Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(76)A Acc72K(I)Data8(7	255 bytes Acc722K[I],Data8[522],a Communication 15516 bytes Acc722K[I],Data8[520],a Hi HANDEAKE CHANNEL 16 BTS Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[522],a Horogenenable Logic Controller (Pic) 0 CONTROL 8 bytes Acc72K[I],Data8[768],a OUT (Hots System to netX) DPM (Dua1-Port Memory) UNCONTROLED COMMON STATUS 64 bytes CONTROLED COMMON STATUS 64 bytes	256 bytes Acc72EV(1)Data8[512].a Communication 15616 bytes Acc72EV(1)Data8[768].a IN HANDSHAC CHANNEL 16 BTS Acc72EV(1)Data8[520].a Acc72EV(1)Data8[520].a Acc72EV(1)Data8[520].a Acc72EV(1)Data8[520].a Acc72EV(1)Data8[520].a Acc72EV(1)Data8[768].a Acc72EV(1)Data8[768].a OUT (Host System to merk) DPM (Dua1Pot Memory) UNCONTROLLED 0 COMMON STATUS 64 byte Acc72EV(1)Data8[768].a Acc72EV(1)Data8[768].a Acc72EV(1)Data8[768].a Acc72EV(1)Data8[768].a Acc72EV(1)Data8[768].a Acc72EV(1)Data8[768].a OUT (Host System to merk) DPM (Dua1Pot Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address New Hondhake Register Communication Register Nost Handshake Register Communication Class Conformance Class Number of Subblocks Size Start Offset Transfer Type Handshake Mode Handshake Mode Handshake Mode Handshake Mode Handshake Rit Size Start Offset Transfer Direction Transfer Direction Tr	256 bytes Acc72EX[i].Data8[512].a Communication 150516 bytes Acc72EX[i].Data8[576].a Ri HANDSHAKE (GANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a MATER Managing Node 0 9 COMTROL 8 bytes Ac72EX[i].Data8[768].a OUTICOL 9 COMMON STATUS 6 bytes Ac72EX[i].Data8[768].a OUTICOLED 0 COMMON STATUS 6 bytes Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a OUTICOLTROLLED 0 COMMON STATUS 6 bytes Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a Ac72EX[i].Data8[768].a	256 bytes Acc722K[I]Data8[512].a Communication 7424 bytes Acc722K[I]Data8[768].a IN HANDSHAK CHANNEL 16 BITS Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc72K[I]Data8[520].a COMTROL 8 bytes Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a COMMON STATUS G4 bytes Acc72K[I]Data8[784].a IN (net to Host System) DPM (Dua1671 Memory) UNCONTROLLED 0 EXTENDED STATUS	256 bytes Acc7220(J).Data8[512].a Communication 15616 bytes Acc7220(J).Data8[520].a Acc7220(J).Data8[520].a Acc7223(J).Data8[520].a Acc7223(J).Data8[520].a Acc7223(J).Data8[520].a Acc7223(J).Data8[520].a CONTROL B bytes Acc7223(J).Data8[768].a OUT (Host System to netX) DPM (DuaFArt Memory) UNCONTROLED O COMMON STATUS 64 bytes Acc7223(J).Data8[784].a N (retX to Host System) OPM (DuaFArt Memory) UNCONTROLED O COMMON STATUS 64 bytes Acc7223(J).Data8[784].a N (retX to Host System) OPM (DuaFArt Memory) UNCONTROLED O EXTENDED STATUS 42 bytes Acc7225(J).Data8[888].a N (retX to Host System)	256 bytes Acc7225(J],Data8[512],a Communication 7424 bytes Acc7225(J],Data8[52],a IN HANDSHKE CHANNEL 16 BITS Acc7225(J],Data8[52],a Acc7225(J],Data8[52],a SLVE 0 9 9 CONTROL 8 bytes Acc7254(J],Data8[52],a SLVE 0 9 CONTROL 8 bytes Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a Acc7254(J],Data8[768],a N, Inext to Host System) DPM (Dua1450rt Memory) UNCONTROLED 0 0 EXTENDED STATUS	256 bytes Acc722K[I].bata8[52].a Communication 15516 bytes Acc722K[I].bata8[52].a MATER IN HANDSHAKE CHANNEL 16 BITS Acc722K[I].bata8[52].a Acc722K[I].bata8[52].a MASTER 9 CONTROL 8 bytes Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a Acc72K[I].bata8[578].a NCONTROLLED 0 COMMON STATUS 64 bytes Acc72K[I].bata8[578].a NCONTROLLED 0 EXTENDED STATUS	256 bytes Acc72EX[I].Data8[512].a Communication 7424 bytes Acc72EX[I].Data8[768].a IN HANDSHAK CHANNEL 16 BTS Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a SchWE Scanner 0 9 CONTROL 8 bytes Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[768].a Acc72EX[I].Data8[764].a N(neX to Host System) DPM (Dua167 Memory) UNCONTROLLED 0 EXTENDED STATUS	256 bytes Acc725K(I].Data8[512].a Communication 7424 bytes Acc725K(I].Data8[512].a IN HANGKAC GHANNEL 16 BITS Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a Acc725K(I].Data8[522].a SLAVE 0 9 9 CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 CONTROL 8 bytes CONTROL 9 CONTROL	256 bytes Ac:22EX[(].Data8[512].a Communication 156 Ib bytes Ac:72EX[(].Data8[52].a IN HANGSKRC CHANNEL 16 BITS Ac:72EX[(].Data8[52].a MASTER IN CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 CONTROL 9	256 bytes Acc722K[I]Data8[512].a Communication 15616 bytes Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a Acc722K[I]Data8[520].a SLWE 0 9 9 CONTROL 8 bytes Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a Acc72K[I]Data8[768].a COMMON STATUS 64 bytes Acc72K[I]Data8[764].a IN (netX to Host System) DPM (Data8[764].a IN (NetX to Host System	256 bytes Acc722C(J) Data8[512].a Communication 15616 bytes Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a SCANNEC 0 9 9 CONTROL 8 bytes Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[520].a Acc722C(J) Data8[576].a O COMMON STATUS 64 bytes Acc722C(J) Data8[576].a N(netX to Hood System) DPM (Data767 Memory) UNCONTROLLED 0 EXTENDED STATUS	256 bytes Acc7228(JI).Data8[512].a Communication 15616 bytes Acc7228(JI).Data8[520].a Acc7228(JI).Data8[520].a Acc7228(JI).Data8[520].a Acc7228(JI).Data8[520].a Acc7228(JI).Data8[520].a Acc728(JI).Data8[768].a OUTROL 8 bytes Acc728(JI).Data8[768].a OUTROL 9 CONTROL 8 bytes Acc728(JI).Data8[768].a OUT (MoX 5ystem to netX) DPM (Dua1476rt Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc728(JI).Data8[768].a Nf (ReX to fock 5ystem) 0 COMMON STATUS 64 bytes Acc728(JI).Data8[768].a Nf (ReX to fock 5ystem) 0 COMMON STATUS 64 bytes Acc728(JI).Data8[768].a Nf (ReX to fock 5ystem) 0 COMMON STATUS 64 bytes Acc728(JI).Data8[848].a Nf (ReX to fock 5ystem) 0 COMMON STATUS 64 bytes Acc728(JI).Data8[848].a Nf (ReX to fock 5ystem)	256 bytes Acc722K(I)Data8(512).a Communication 15616 bytes Acc722K(I)Data8(512).a IN HANGKAC CHANNEL 16 BITS Acc722K(I)Data8(520).a Acc722K(I)Data8(520).a Acc722K(I)Data8(520).a Acc722K(I)Data8(520).a Acc722K(I)Data8(520).a MESSAGING Combination Firmware 0 9 CONTROL 8 bytes Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a Acc72K(I)(Data8(768).a N(MACONTROLLED 0 COMMON STATUS 66 bytes Acc72K(I)Data8(784).a IN(neX to Host system) DPM (DatA9CT Memory) UNCONTROLED 0 EXTENDED STATUS	255 bytes Acc722K[I],Data8[512].a Communication 15516 bytes Acc722K[I],Data8[520].a Acc722K[I],Data8[520].a Acc722K[I],Data8[22].a Ino-CONTROLES Programmable Logic Controller (Pic) 0 9 CONTROL 8 bytes Acc722K[I],Data8[76].a Acc722K[I],Data8[76].a Acc722K[I],Data8[76].a Acc722K[I],Data8[76].a Acc722K[I],Data8[76].a Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Downorthe Acc72K[I],Data8[76].a Acc72K[I],Data8[76].a Downorthe Acc72K[I],Data8[76].a INCONTROLED 0 COMMON STATUS 64 bytes Acc72K[I],Data8[76].a IN(netX to Host System) DPM (Dual-Port Memory) UNCONTROLED 0 EXTENDED STATUS	256 bytes Acc72EV(i)Data8[512].a Communication 15616 bytes Acc72EV(i)Data8[768].a IN HANDSHAKE GHANNEL 16 BITS Acc72EV(i)Data8[520].a Acc72EV(i)Data8[520].a Acc72EV(i)Data8[520].a Acc72EV(i)Data8[520].a Acc72EV(i)Data8[520].a Acc72EV(i)Data8[768].a COMTROL 8 bytes Acc72EV(i)Data8[768].a COMTROL 8 bytes Acc72EV(i)Data8[768].a DPM (bute)Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EV(i)Data8[784].a N (nek to Host System) DPM (bute)Port Memory) UNCONTROLED 0 EXTENDED STATUS
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Size of Channel Channel Start Address Pacition of Handhake Cells Size of Handhake Cells Size of Handhake Cells North Handhake Register Communication Cilsas Protocol Class Conformance Class Number of Subblocks Subblock 0 Size Start Offset Transfer Type Handhake Mode Handhake Mode Handhake Bit Subblock 1 Size Size Start Offset Transfer Direction Transfer Type Handhake Bit Subblock 2 Size Start Offset Transfer Direction Transfer Direction Transfer Size Start Offset Transfer Direction Transfer Size Start Offset Transfer Direction Transfer Type Handhake Bit	256 bytes Acc72EX[i].Data8[512].a Communication 155616 bytes Acc72EX[i].Data8[576].a NH KANDSHAKE (GANNELL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a MATSER Managing Node 0 9 CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT (Host System to netX) DVM (Da4Port Memory) UNCONTROLLE 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a N (netX to Host System) DVM (Du4Port Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes Acc72EX[i].Data8[848].a N (netX to Host System) DVM (Nu4Port Memory)	256 bytes Acc722K[I].Data8[512].a Communication 7424 bytes Acc722K[I].Data8[512].a IN HANDSHAK CHANNEL 16 BITS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a CONTROL 8 bytes Acc722K[I].Data8[768].a OUT (Host System to netX) DPM (DauHort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a N(netX to Host System) DPM (DauHort Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes Acc722K[I].Data8[848].a N(netX to Host System) DPM (DauHort Memory) UNCONTROLLED	255 bytes Acc722C(J)[Data8[512] a Communication 1556 bytes Acc722C(J)[Data8[522] a MASTER Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[763] a OUT (Mac 5) stem to netX) DPM (Dual 40 tr Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc720[D][Data8[764] a IN (netXto Hois System) DPM (Dual 40 tr Memory) UNCONTROLLED 0 EXTENDED STATUS 42 bytes Acc722C(J][Data8[548] a IN (netXto Hois System) DPM (Dual 40 tr Memory) UNCONTROLLED 0	256 bytes Acc728(F)[Data8[512] a Communication 7424 bytes Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Server 0 9 CONTROL 8 bytes Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 COMMON STATUS 64 bytes Acc728(F)[Data8[784] a N(netX to Hoct System) 0 COMMON STATUS 64 bytes Acc728(F)[Data8[784] a N(netX to Hoct System) 0 COMMON STATUS 64 bytes Acc728(F)[Data8[848] a N(netX to Hoct System) 0 COMMON STATUS 64 bytes Acc728(F)[Data8[848] a N(netX to Hoct System) 0 CMICONTROLLED CMICONT	256 bytes Acc722K[i].bata8[52].a Communication 15516 bytes Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a MASTER Scanner 0 9 CONTROL 8 bytes Acc72K[i].bata8[768].a OUT (Hots System to netX) DFM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72K[i].bata8[78].a NCONTROLED 0 COMMON STATUS 64 bytes Acc72K[i].bata8[78].a NCONTROLED 0 COMMON STATUS 64 bytes Acc72K[i].bata8[78].a NCONTROLED 0 COMMON STATUS 54 bytes Acc72K[i].bata8[78].a NCONTROLED 0 EXTENDED STATUS 432 bytes Acc72K[i].bata8[78].a N(netX to Hots System) DFM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72K[i].bata8[78].a N(netX to Hots System) DFM (Dual-Port Memory) DFM (Dual-Port Memory) DFM (Dual-Port Memory) DFM (Dual-Port Memory)	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[512].a IN HANDSHAK CHANNEL 16 BTS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a SLAVE Scanner 0 9 CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT (Host System to net.X) DPM (DuaHOrt Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[784].a IN (net to Host System) DPM (DuaHOrt Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes Acc72EX[i].Data8[848].a IN (net Xto Host System) DPM (DuaHOrt Memory)	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[512].a IN HANDSHAC CHANNEL 16 BITS Acc72EX[i].Data8[522].a SLAVE 0 9 CONTROL 8 bytes Acc72EX[i].Data8[522].a SLAVE 0 9 CONTROL 8 bytes Acc72EX[i].Data8[52].a DUT [Hots System to netX] DPM [DataPort Memory] UNCONTROLED 0 COMMON STATUS 66 bytes Acc72EX[i].Data8[78].a IN (neXto Hots System) DPM (DataPort Memory) UNCONTROLED 0 COMMON STATUS 66 bytes Acc72EX[i].Data8[78].a IN (neXto Hots System) DPM (DataPort Memory) UNCONTROLED 0 CXTENDED STATUS 422 bytes Acc72EX[i].Data8[848].a IN (neXto Hots System) DPM (DataPort Memory) DPM (DataPort Memory) DPM (DataPort Memory) DPM (DataPort Memory) DPM (DataPort Memory)	256 bytes Acc722K(I)Data8[512] a Communication 15616 bytes Acc722K(I)Data8[52] a MATER BITS Acc722K(I)Data8[52] a Acc722K(I)Data8[52] a Acc722K(I)Data8[52] a MATER bController 0 9 CONTROL 8 bytes Acc722K(I)Data8[52] a MATER Acc722K(I)Data8[52] a MATER CONTROL 8 bytes Acc72K(I)Data8[52] a DUT (Hota System to netX) DVM (Data97ct Memory) UNCONTROLLED 0 COMMON STATUS 66 bytes Acc72K(I)Data8[78] a N(netXto Hota System) DVM (Data97ct Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc72K(I)Data8[78] a N(netXto Hota System) DVM (Data97ct Memory) DVM (Da	256 bytes Acc728(J] Data8[512] a Communication 15616 bytes Acc728(J] Data8[52] a Acc728(J] Data8[52] a Acc728(J] Data8[52] a Acc728(J] Data8[52] a Acc728(J] Data8[52] a SLAVE 0 9 9 CONTROL 8 bytes Acc728(J] Data8[52] a Acc728(J] Data8[52] a CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 0 COMMON STATUS 64 bytes Acc728(J] Data8[78] a IN (netXto Hock System) 0 COMMON STATUS 64 bytes Acc728(J] Data8[78] a IN (netXto Hock System) 0 COMMON STATUS 64 bytes Acc728(J] Data8[78] a IN (netXto Hock System) 0 COMMON STATUS 64 bytes Acc728(J] Data8[84] a IN (netXto Hock System) 0 COMMON STATUS 64 bytes Acc728(J] Data8[84] a IN (netXto Hock System) 0 CMMON STATUS 64 bytes Acc728(J] Data8[84] a 0 CMMON STATUS 64 bytes Acc728(J] Data8[84] a 10 Acc728(J]	256 bytes Acc722 C(I) Data8[512].a Communication 1556 bytes Acc722 C(I) Data8[52].a IN HANSHKK CHANNEL 16 BITS Acc728 (I) Data8[52].a Acc728 (I) Data8[52].a SCANNER b Obvice 0 9 CONTROL 8 bytes Acc722 (I) Data8[76].a OUT (MoL 8 bytes Acc722 (I) Data8[76].a OUT (MoL 9 CONTROL 8 bytes CONTROL 8 bytes CONTROL 10 Dovice 0 0 COMMON STATUS 64 bytes Acc722 (I) Data8[76].a OUT (MoL 10 Dovice 0 0 CONMON STATUS 64 bytes Acc728 (I) Data8[76].a IN (NetX to Hoc System to netX) 0 CONMON STATUS 64 bytes Acc728 (I) Data8[76].a IN (NetX to Hoc System to netX) 0 CONMON STATUS 64 bytes Acc728 (I) Data8[76].a IN (NetX to Hoc System) 0 CONMON STATUS 64 bytes Acc728 (I) Data8[76].a IN (NetX to Hoc System) 0 CONMON STATUS 64 bytes Acc728 (I) Data8[76].a IN (NetX to Hoc System) 0 PM (Dual-Port Memory) DM (Dual-Port Memory) DM (Dual-Port Memory) DM (Dual-Port Memory) DM (Dual-Port Memory)	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Downer 0 CONTROL 8 bytes Acc722K[I].Data8[768].a DUT (IndS System to netX) DPM (Dual+Ort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a N (netX to Hox System) DPM (Dual+Dort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a N (netX to Hox System) DPM (Dual+Dort Memory) UNCONTROLLED 0 CXTENDED STATUS 432 bytes Acc722K[I].Data8[548].a N (netX to Hox System) DPM (Dual+Dort Memory) DPM (Dual+Dort Memory)	256 bytes Acc722K(i)Data8(512) a Communication 15616 bytes Acc722K(i)Data8(512) a Acc722K(i)Data8(512) a Acc722K(i)Data8(52)] a Acc722K(i)Data8(52)] a Acc722K(i)Data8(52)] a Acc722K(i)Data8(52)] a Acc722K(i)Data8(52) a MESSAGNG Combination Firmware 0 9 CONTROL 8 bytes Acc722K(i)Data8(78)] a OUT (Hots System to netX) DPM (Dal-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72K(i)Data8(78)] a IN (netX to Host System) DPM (Dal-Port Memory) UNCONTROLED 0 EXTENDED STATUS 422 bytes Acc722K(i)Data8(78)] a IN (netX to Host System) DPM (Dal-Port Memory) UNCONTROLED 0 EXTENDED STATUS 422 bytes Acc72K(i)Data8(54)] a IN (netX to Host System) DPM (Dal-Port Memory) UNCONTROLED 0	2 55 bytes Acc722K[I],Data8[52],J Communication 1 5516 bytes Acc722K[I],Data8[520],J NH HANDSHAKC CHANNEL 16 BITS Acc722K[I],Data8[520],J Acc722K[I],Data8[52],J IO-CONTROLES Programmable Logic Controller (PIc) 0 9 CONTROL 8 bytes Acc722K[I],Data8[768],J OUT (Intex5 System to netX) DWI (Du4Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K[I],Data8[764],J NICREX ID 405 System) DVI (Intex Lin Host System)	256 bytes Acc72EX[i].Data8[512] a Communication 15616 bytes Acc72EX[i].Data8[768] a IN HANDSHAKE CHANNEL 16 BITS Acc72EX[i].Data8[520] a Acc72EX[i].Data8[520] a Acc72EX[i].Data8[520] a Acc72EX[i].Data8[520] a Acc72EX[i].Data8[520] a CONTROL 8 bytes Acc72EX[i].Data8[768] a OUT [HotS system to netX] DFM (Dua4Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[784] a N (netX to HotS System) DFM (Dua4Port Memory) UNCONTROLED 0 EXTENDED STATUS 43 bytes Acc72EX[i].Data8[784] a N (netX to HotS System) DFM (Dua4Port Memory) UNCONTROLED 0 EXTENDED STATUS 43 bytes Acc72EX[i].Data8[648] a N (netX to HotS System)
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Channel Start Address Position of Handshake Cells Size of Handshake Cells NetX Handshake Cells Size of Handshake Cells NetX Handshake Register Host Handshake Register Communication Class Protocol Class Conformance Class Number of Subbocks Size Size Oftest Transfer Direction Transfer Direction Transfer Direction Transfer Spreach Handshake Bit Subbock 1 Size Start Offest Transfer Type Handshake Bit Subbock 1 Size Start Offest Transfer Type Handshake Bit Subbock 1 Size Start Offest Transfer Direction Transfer Direction	256 bytes Acc72EX[I].Data8[512].a Communication 15616 bytes Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[768].a OUT (Host System to netX) DPM (Out-Port Memory) UNCONTROLLED O COMMON STATUS 64 bytes Acc72EX[I].Data8[784].a NorthColl.Bata8[784].a Acc72EX[I].Data8[784].a Acc72EX[I].Data8[784].a NorthColl.Bata8[784].a Acc72EX[I].Data8[784].a NorthColl.Bata8[784].a	256 bytes Acc728([i].Data8[512].a Communication 7424 bytes Acc728([i].Data8[52].a IN HANDSHAC CHANNEL 16 BTS Acc728([i].Data8[520].a Acc728([i].Data8[520].a Acc728([i].Data8[520].a Acc728([i].Data8[52].a Managing Node 0 9 CONTROL 8 bytes Acc728([i].Data8[768].a OUT (Host System to netX) DPM (Dua1-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc728([i].Data8[84].a N (netX to Ads System) D (Data8[54].a	256 bytes Acc7220(J).Data8[512].a Communication 15616 bytes Acc7220(J).Data8[520].a Acc7220(J).Data8[520].a Acc7223(J).Data8[520].a Acc7223(J).Data8[520].a Acc7223(J).Data8[520].a Acc7223(J).Data8[520].a CONTROL B bytes Acc7223(J).Data8[768].a OUT (Host System to netX) DPM (DuaFArt Memory) UNCONTROLED O COMMON STATUS 64 bytes Acc7223(J).Data8[784].a N (retX to Host System) OPM (DuaFArt Memory) UNCONTROLED O COMMON STATUS 64 bytes Acc7223(J).Data8[784].a N (retX to Host System) OPM (DuaFArt Memory) UNCONTROLED O EXTENDED STATUS 42 bytes Acc7225(J).Data8[888].a N (retX to Host System)	256 bytes Acc7225(J],Data8[512],a Communication 7424 bytes N HANDSHARC CHANNE 16 BITS Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[522],a Sever 0 9 CONTROL 8 bytes Acc7225(J],Data8[768],a OUT (Host System to netX) DPM (Dua1676 M temory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc7225(J],Data8[784],a NDF(Dua1676 M temory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc7225(J],Data8[784],a NDF(Dua1676 M temory) UNCONTROLED 0 CTMMON STATUS 64 bytes Acc7225(J],Data8[784],a NDF(Dua1676 M temory) UNCONTROLED 0 EXTENDED STATUS 422 bytes Acc7225(J],Data8[848],a N (NetX to 465 System)	256 bytes Acc722K[i].bata8[512].a Communication 15516 bytes Acc722K[i].bata8[576].a Acc722K[i].bata8[520].a Acc722K[i].bata8[520].a Acc722K[i].bata8[520].a Acc722K[i].bata8[52].a Mca12K CONTROL 8 bytes Acc722K[i].bata8[768].a OUT (Host System to netX) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K[i].bata8[768].a COMMON STATUS 64 bytes Acc722K[i].bata8[768].a COMMON STATUS 64 bytes Acc72K[i].bata8[768].a Acc72K[i].bata8[768].a Acc72K[i].bata8[768].a Acc72K[i].bata8[768].a Acc72K[i].bata8[768].a Acc72K[i].bata8[848].a Acc72K[i].bata8[848].a Acc72K[i].bata8[848].a Acc72K[i].bata8[848].a Acc72K[i].bata8[848].a Acc72K[i].bata8[848].a Acc72K[i].bata8[848].a Acc72K[i].bata8[848].a	256 bytes Acc72EV(I),Data8[512],a Communication 7424 bytes Acc72EV(I),Data8[520],a Acc72EV(I),Data8[520],a Acc72EV(I),Data8[520],a Acc72EV(I),Data8[520],a Acc72EV(I),Data8[520],a Acc72EV(I),Data8[520],a Acc72EV(I),Data8[548],a OUT (Host System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0 COMMON STATUS E4 bytes Acc72EV(I),Data8[548],a NCHARD STATUS E4 bytes Acc72EV(I),Data8[548],a NCHARD STATUS E4 bytes Acc72EV(I),Data8[548],a NC NOTROLLED 0 DEXTENDED STATUS E42 bytes Acc72EV(I),Data8[548],a NC NOTROLLED 0	256 bytes Acc72EX[[]Data8[512],a Communication 724 bytes Acc72EX[[]Data8[528],a Ni HAUSERK CHANNEL 16 BITS Acc72EX[[]Data8[520],a Acc72EX[[]Data8[520],a Acc72EX[]]Data8[520],a Acc72EX[]Data8[520],a Acc72EX[]Data8[520],a Acc72EX[]Data8[520],a Acc72EX[]Data8[520],a CONTROL 8 bytes Acc72EX[]Data8[528],a OUT (Host System to netX) DPM (Dua1-Port Memory) UNCONTROLLED 0 COMMON STATUS 66 bytes Acc72EX[]Data8[528],a Acc72EX[]Data8[528],a Acc72EX[]Data8[528],a Acc72EX[]Data8[528],a NCONTROLLED 0 EXTENDED STATUS 42 bytes Acc72EX[]Data8[84],a Ni NetX Host System)	256 bytes Ac:72EX(I)Data8[512].a Communication 15616 bytes Ac:72EX(I)Data8[512].a Nr HAUSEARC CHANNEL 16 BITS Ac:72EX(I)Data8[520].a Ac:72EX(I)Data8[520].a Ac:72EX(I)Data8[522].a MASTEN b COMTROL 8 bytes Ac:72EX(I)Data8[76].a OUT (Host System to netX) DPM (Dua140-rt Memory) UNCONTROLED 0 COMMON STATUS 56 bytes Ac:72EX(I)Data8[76].a Ac:72EX(I)Data8[76].a COMMON STATUS 56 bytes Ac:72EX(I)Data8[76].a Ac:72EX(I)Data8[76].	256 bytes Acc7225(J].Data8[512].a Communication 15616 bytes Acc7225(J].Data8[52].a IN HANDSHAC CHANNEL 16 BITS Acc725(J].Data8[520].a Acc725(J].Data8[522].a 9 CONTROL 8 bytes Acc725(J].Data8[576].a OUT (Host System to netX) DPM (Dua1676 Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc725(J].Data8[78].a Acc725(J].Data8[78].a North (Dua1676 Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc7252(J].Data8[78].a North (Dua1676 Memory) UNCONTROLED 0 CXTUDED STATUS 42 bytes Acc7225(J].Data8[84].a Ni NietXtok System to Acc725(J].Data8[84].a Ni NietXtok System)	256 bytes Acc722 EX[I].Data8[512].a Communication 15616 bytes Acc722 EX[I].Data8[520].a Acc722 X[I].Data8[520].a Acc723 X[I].Data8[520].a Acc723 X[I].Data8[520].a Acc723 X[I].Data8[520].a Acc72 X[I].Data8[520].a CONTROL B bytes Acc728 X[I].Data8[768].a OUT (Host System to netX) DPM (Dua1-Port Memory) UNCONTROLLED 0 COMMON STATUS 56 bytes Acc728 X[I].Data8[768].a N (netX) to Ass System) DPM (Dua1-Port Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes Acc722 X[I].Data8[84].a N (netX) to Ass System)	256 bytes Acc7228(JI).Data8[512].a Communication 15616 bytes Acc7228(JI).Data8[520].a Acc7228(JI).Data8[520].a Acc7228(JI).Data8[520].a Acc7228(JI).Data8[520].a Acc7228(JI).Data8[520].a Acc728(JI).Data8[768].a OUTROL 8 bytes Acc728(JI).Data8[768].a OUTROL 9 CONTROL 8 bytes Acc728(JI).Data8[768].a OUT (MoX 5ystem to netX) DPM (Dua1476rt Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc728(JI).Data8[768].a Nf (ReX to fock 5ystem) 0 COMMON STATUS 64 bytes Acc728(JI).Data8[768].a Nf (ReX to fock 5ystem) 0 COMMON STATUS 64 bytes Acc728(JI).Data8[768].a Nf (ReX to fock 5ystem) 0 COMMON STATUS 64 bytes Acc728(JI).Data8[848].a Nf (ReX to fock 5ystem) 0 COMMON STATUS 64 bytes Acc728(JI).Data8[848].a Nf (ReX to fock 5ystem)	256 bytes Acc722K(i)Data8(512)a Communication 15616 bytes Acc722K(i)Data8(52)a Acc722K(i)Data8(52)a Acc722K(i)Data8(52)a Acc722K(i)Data8(52)a ComtRoL 8 bytes CONTROL 8 bytes CONTROL 8 bytes Acc722K(i)Data8(768)a OUT (Host System to netX) DPM (Dua140-FOT Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K(i)Data8(78)a Acc	2 55 bytes Acc722K[I],Data8[522],a Communication 1 5516 bytes Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[521],a Ho_COMTROLE B bytes Acc72K[I],Data8[768],a OUT (Host System to netX) DPM (Dub4Port Memory) UNCONTROLED COMMON STATUS 64 bytes Acc72K[I],Data8[768],a Acc72K[I],Data8[7	256 bytes Acc72EV(1)Data8[512].a Communication 15616 bytes Acc72EV(1)Data8[768].a IN HANDSHAC CHANNEL 16 BTS Acc72EV(1)Data8[769].a Acc72EV(1)Data8[520].a Acc72EV(1)Data8[520].a Acc72EV(1)Data8[520].a Acc72EV(1)Data8[768].a OUT (Host System to next) DPM (Dala9[768].a OUT (Host System to next) DPM (Dala9[768].a OUT (Host System to next) DPM (Dala9100000000000000000000000000000000000
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Channel Start Address Position of Handchake Cells Size of Handchake Cells Size of Handchake Cells NetX Handchake Register Host Handchake Register Communication Class Protocol Class Conformance Class Number of Subblocks Size Start Offset Transfer Direction Transfer Direction Transfer Direction Transfer Type Handchake Mode Handshake Bit	256 bytes Acc72EX[I].Data8[512].a Communication 1556 bytes Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a B D COMTOD.L B D COMTOD.L B D COMMON STATUS G bytes Acc72EX[I].Data8[78].a DUNCONTROLLED D COMMON STATUS G bytes Acc72EX[I].Data8[78].a N(netXt0 Hots System) DFM (Cu4-Port Memory) UNCONTROLLED 0 CYTINDED STATUS 432 bytes Acc72EX[I].Data8[78].a NCTROLLED D D CYTINDED STATUS 432 bytes Acc72EX[I].Data8[78].a NCTROLLED D D D D D D D D D D D D D D D D D D	256 bytes Acc728(I)[Data8[512].a Communication 7424 bytes Acc728(I)[Data8[522].a Kac728(I)[Data8[520].a Acc728(I)[Data8[520].a Acc728(I)[Data8[520].a Acc728(I)[Data8[520].a Acc728(I)[Data8[520].a Acc728(I)[Data8[520].a B bytes Acc728(I)[Data8[768].a OUT (Hots System to netX) DPM (DatA9rot Memory) UNCONTROLLED 0 EXTENDED STATUS 64 bytes Acc728(I)[Data8[784].a N(InetX to Hots System) DPM (DatA9rot Memory) UNCONTROLLED 0 EXTENDED STATUS 422 bytes Acc728(I)[Data8[784].a N(InetX to Hots System) DPM (DatA9rot Memory) UNCONTROLLED 0 EXTENDED STATUS 422 bytes Acc728(I)[Data8[848].a N (InetX to Hots System) DPM (DatA9rot Memory) UNCONTROLLED 0 EXTENDED STATUS	256 bytes Acc722(J)[Data8[512].a Communication 15616 bytes Acc722[J][Data8[582].a IN HANDSHK CHANNEL 16 BITS Acc722[J][Data8[520].a Acc722[J][Data8[520].a Acc722[J][Data8[520].a Acc722[J][Data8[520].a MASTE Berver 0 9 9 COMTROL 8 bytes Acc722[J][Data8[768].a OUT (Host System to netX) DPM (Dua1+Dort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722[J][Data8[784].a IN (netX to Not System) DPM (Dua1+Dort Memory) UNCONTROLED 0 EXTENDED STATUS 42 bytes Acc722[J][Data8[784].a IN (netX to Not System) DPM (Dua1+Dort Memory) UNCONTROLED 0 EXTENDED STATUS 42 bytes Acc722[J][Data8[784].a IN (netX to Not System) DPM (Dua1+Dort Memory) UNCONTROLED 0 EXTENDED STATUS 42 bytes Acc722[J][Data8[784].a IN (netX to Not System) DPM (Dua1+Dort Memory) UNCONTROLED 0	256 bytes Acc7225(J],Data8[512],a Communication 7424 bytes Acc7225(J],Data8[512],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a CONTROL 8 bytes Acc725(J],Data8[520],a CONTROL 9 0 CONTROL 8 bytes Acc725(J],Data8[768],a OUT (Host System to netX) DPM (Dua1-bort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc725(J],Data8[784],a N (netX to Host System) DPM (Dua1-bort Memory) UNCONTROLLED 0 CONTROLLED 0 CONTROLLED 0	256 bytes Acc722K[i].bata8[512].a Communication 15516 bytes Acc722K[i].bata8[520].a Acc722K[i].bata8[520].a Acc722K[i].bata8[520].a Acc722K[i].bata8[520].a Acc722K[i].bata8[52].a MASTE Scanner 0 9 9 9 COMMON TATOL 8 bytes Acc722K[i].bata8[768].a OUT (Hox System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0 COMMON STATUS 66 bytes Acc722K[i].bata8[768].a IN (netX to Hox System) DPM (Dual-Port Memory) UNCONTROLLED 0 EXTENDED STATUS 423 bytes Acc722K[i].bata8[784].a IN (netX to Hox System) DPM (Dual-Port Memory) UNCONTROLLED 0	256 bytes Acc72EX[I],Data8[512],a Communication 7424 bytes Acc72EX[I],Data8[768],a IN HANDSHAKE CHANNEL 16 BTS Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Bytes Acc72EX[I],Data8[768],a OUT (Hots System to netX) DPM (DuaHort Memory) UNCONTROLLED 0 EXTENDED STATUS 64 bytes Acc72EX[I],Data8[784],a N (netX to Hots System) DPM (DuaHort Memory) UNCONTROLLED 0 EXTENDED STATUS 64 bytes Acc72EX[I],Data8[848],a N (netX to Hots System) DPM (DuaHort Memory) UNCONTROLLED 0 EXTENDED STATUS 64 bytes Acc72EX[I],Data8[848],a N (netX to Hots System) DPM (DuaHort Memory) UNCONTROLLED 0	256 bytes Acc725K(I].Data8[512].a Communication 7424 bytes Acc725K(I].Data8[512].a IN HAUGENKE CHANNEL I & BITS Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a SLAVE 0 9 9 COMTROL B bytes Acc725K(I].Data8[768].a OUT [Host System to netX) DPM (Dua14Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc725K(I].Data8[764].a IN (netX to Host System) DPM (Dua14Port Memory) UNCONTROLLED 0 EXTENDED STATUS 42 bytes Acc725K(I].Data8[764].a IN (netX to Host System) DPM (Dua14Port Memory) UNCONTROLLED 0	25 bytes Ac:72EX(I)Data8[512].a Communication 15616 bytes Ac:72EX(I)Data8[52].a Ni HAUGHKK CHANNEL 16 BITS Ac:72EX(I)Data8[52].a MASTEM b Controller 0 9 9 0 COMTROL 8 bytes Ac:72EX(I)Data8[52].a MASTEM b Controller 0 9 0 COMTROL 8 bytes Ac:72EX(I)Data8[58].a NUCCONTROLED 0 COMMON STATUS 66 bytes Ac:72EX(I)Data8[58].a NI (neX to hick System to neX) 0 DPM (Dual-Port Memory) UWCONTROLED 0 COMMON STATUS 64 bytes Ac:72EX(I)Data8[58].a NI (neX to hick System) DPM (Dual-Port Memory) UWCONTROLED 0 COMMON STATUS 64 bytes Ac:72EX(I)Data8[58].a NI (neX to hick System) DPM (Dual-Port Memory) UWCONTROLED 0 COMMON STATUS 64 bytes Ac:72EX(I)Data8[84].a NI (neX to hick System) DPM (Dual-Port Memory) UWCONTROLED 0 C	256 bytes Acc722K[I]Data8[512]a Communication 1556 bytes Acc722K[I]Data8[520]a Acc722K[I]Data8[520]a Acc722K[I]Data8[520]a Acc722K[I]Data8[520]a Acc722K[I]Data8[520]a CONTROL Bytes Acc722K[I]Data8[520]a CONTROL Bytes Acc722K[I]Data8[526]a OUT (Host System to netX) DPM (Dua14Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K[I]Data8[784]a IN (InetX to Host System) DPM (Dua14Fort Memory) UNCONTROLED 0 CONTROLED 0 EXTENDED STATUS 42 bytes Acc722K[I]Data8[784]a IN (InetX to Host System) DPM (Dua14Fort Memory) UNCONTROLED 0 EXTENDED STATUS 42 bytes Acc722K[I]Data8[84]a IN (InetX to Host System) DPM (Dua14Fort Memory) UNCONTROLED 0 EXTENDED STATUS 42 bytes Acc722K[I]Data8[84]a IN (InetX to Host System) DPM (Dua14Fort Memory) UNCONTROLED 0 EXTENDED STATUS 42 bytes Acc722K[I]Data8[84]a IN (InetX to Host System) DPM (Dua14Fort Memory) UNCONTROLED 0 EXTENDED STATUS	256 bytes Acc722 Cf(I)Data8[512].a Communication 15616 bytes Acc722 Rf(I)Data8[512].a IN HANDSHKC CHANNEL 16 BITS Acc722 Rf(I)Data8[520].a Acc722 Rf(I)Data8[520].a Acc722 Rf(I)Data8[520].a Acc722 Rf(I)Data8[520].a CONTROL 8 bytes Acc72 Rf(I)Data8[768].a OUT (Host System to netX) DPM (Dua1+Dort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722 Rf(I)Data8[784].a IN (netX to Nois System) DPM (Dua1+Dort Memory) UNCONTROLED 0 CCNMON STATUS 64 bytes Acc722 Rf(I)Data8[784].a IN (netX to Nois System) DPM (Dua1+Dort Memory) UNCONTROLED 0 CCNMON STATUS 64 bytes Acc722 Rf(I)Data8[784].a IN (netX to Nois System) DPM (Dua1+Dort Memory) UNCONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED CONTR	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Data72 CONTROL B bytes Acc72K[I].Data8[765].a OUT (Host System to netX) DPM (Dua1Fort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a N (netX to Host System) DPM (Dua1Fort Memory) UNCONTROLLED 0 C	256 bytes Acc722K(I)Data8(512).a Communication 15616 bytes Acc722K(I)Data8(52).a Acc722K(I)Data8(52).a Acc722K(I)Data8(52).a Acc722K(I)Data8(52).a Acc722K(I)Data8(52).a MisSAGNO Combination Firmware 0 9 CONTROL 8 bytes Acc722K(I)Data8(78).a DUT (Host System to netX) DPM (DuaHoPt Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K(I)Data8(78).a NI (netX to Host System) DPM (DuaHoPt Memory) UNCONTROLLED 0 EXTENDED STATUS 42 bytes Acc722K(I)Data8(8).a NI (netX to Host System) DPM (DuaHoPt Memory) UNCONTROLLED 0 EXTENDED STATUS 42 bytes Acc722K(I)Data8(84).a NI (netX to Host System) DPM (DuaHoPt Memory) UNCONTROLLED 0 EXTENDED STATUS 42 bytes Acc722K(I)Data8(84).a NI (netX to Host System) DPM (DuaHoPt Memory) UNCONTROLLED 0	256 bytes Acc722K[I],Data8[512],a Communication 15516 bytes Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[522],a Do-CONTROL Bytes Control Control Bytes Acc722K[I],Data8[768],a OUT (Hots System to netX) DPM (Dua1+ort Memory) UNCONTROLED O COMMON STATUS G4 bytes Acc722K[I],Data8[768],a OUT (Hots System to netX) DPM (Dua1+Ort Memory) UNCONTROLED O COMMON STATUS G4 bytes Acc722K[I],Data8[784],a Ni (netX to Hot System) DPM (Dua1+Ort Memory) UNCONTROLED O EXTENDED STATUS 422 bytes Acc722K[I],Data8[984],a Ni (netK Hot System) DPM (Dua1+Ort Memory) UNCONTROLED O	256 bytes Acc72EX(I).Data8[512].a Communication 15616 bytes Acc72EX(I).Data8[522].a IN HANDSHAKE GHANNEL 16 BITS Acc72EX(I).Data8[522].a Acc72EX(I).Data8[522].a IX-DEVICE Programmable Logic Controller 67 9 CONTROL 8 bytes Acc72EX(I).Data8[768].a OUT (Host System to netX) DPM (Dua1Port Memory) UNCONTROLLED 0 COMMON STATUS 66 bytes Acc72EX(I).Data8[78].a IN (netX to Host System) DPM (Dua1Port Memory) UNCONTROLLED 0 EXTENDED STATUS 64 bytes Acc72EX(I).Data8[78].a IN (netX to Host System) DPM (Dua1Port Memory) UNCONTROLLED 0
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Size of Channel Channel Start Address Position of Handshake Cells Size of Aradhake Cells Size of Aradhake Cells Nett Mandshake Register Host Mandshake Register Host Mandshake Register Conformance Class Number of Subblocks - Subblock 0 Size Size Cass Size Conformance Class Size Start Offset Transfer Type Handshake Node Handshake Node	256 bytes Acc72EX[i].Data8[512].a Communication 155616 bytes Acc72EX[i].Data8[576].a NH KANDSHAKE (GANNELL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a MATSER Managing Node 0 9 CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT (Host System to netX) DVM (Da4Port Memory) UNCONTROLLE 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a Acc72EX[i].Data8[768].a N (netX to Host System) DVM (Du4Port Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes Acc72EX[i].Data8[848].a N (netX to Host System) DVM (Nu4Port Memory)	256 bytes Acc722K[I].Data8[512].a Communication 7424 bytes Acc722K[I].Data8[512].a IN HANDSHAK CHANNEL 16 BITS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a CONTROL 8 bytes Acc722K[I].Data8[768].a OUT (Host System to netX) DPM (DauHort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a N(netX to Host System) DPM (DauHort Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes Acc722K[I].Data8[848].a N(netX to Host System) DPM (DauHort Memory) UNCONTROLLED	255 bytes Acc722C(J)[Data8[512] a Communication 1556 bytes Acc722C(J)[Data8[522] a MASTER Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[520] a Acc722C(J)[Data8[763] a OUT (Mac 5) stem to netX) DPM (Dual 40 tr Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc720[D][Data8[764] a IN (netXto Hois System) DPM (Dual 40 tr Memory) UNCONTROLLED 0 EXTENDED STATUS 42 bytes Acc722C(J][Data8[548] a IN (netXto Hois System) DPM (Dual 40 tr Memory) UNCONTROLLED 0	256 bytes Acc728(F)[Data8[512] a Communication 7424 bytes Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a Server 0 9 CONTROL 8 bytes Acc728(F)[Data8[520] a Acc728(F)[Data8[520] a CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 COMMON STATUS 64 bytes Acc728(F)[Data8[784] a N(netX to Hoct System) 0 COMMON STATUS 64 bytes Acc728(F)[Data8[784] a N(netX to Hoct System) 0 COMMON STATUS 64 bytes Acc728(F)[Data8[848] a N(netX to Hoct System) 0 COMMON STATUS 64 bytes Acc728(F)[Data8[848] a N(netX to Hoct System) 0 CMICONTROLLED CMICONT	256 bytes Acc722K[i].bata8[52].a Communication 15516 bytes Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a MASTER Scanner 0 9 CONTROL 8 bytes Acc72K[i].bata8[5768].a OUT (Hots System to netX) DFM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 54 bytes Acc72K[i].bata8[78].a NCONTROLED 0 COMMON STATUS 54 bytes Acc72K[i].bata8[78].a NCONTROLED 0 COMMON STATUS 54 bytes Acc72K[i].bata8[78].a NCONTROLED 0 COMMON STATUS 54 bytes Acc72K[i].bata8[78].a NCONTROLED 0 EXTENDED STATUS 432 bytes Acc72K[i].bata8[78].a N(netX to Hots System) DFM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 54 bytes Acc72K[i].bata8[78].a N(netX to Hots System) DFM (Dual-Port Memory) DFM (Dual-Port Memory) DFM (Dual-Port Memory) DFM (Dual-Port Memory)	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[512].a IN HANDSHAK CHANNEL 16 BTS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a SLAVE Scanner 0 9 CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT (Host System to net.X) DPM (DuaHOrt Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[784].a IN (net to Host System) DPM (DuaHOrt Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes Acc72EX[i].Data8[848].a IN (net Xto Host System) DPM (DuaHOrt Memory)	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[512].a IN HANDSHAC CHANNEL 16 BITS Acc72EX[i].Data8[522].a SLAVE 0 9 CONTROL 8 bytes Acc72EX[i].Data8[522].a SLAVE 0 9 CONTROL 8 bytes Acc72EX[i].Data8[52].a DUT [Hots System to netX] DPM [DataPort Memory] UNCONTROLED 0 COMMON STATUS 66 bytes Acc72EX[i].Data8[78].a IN (neXto Hots System) DPM (DataPort Memory) UNCONTROLED 0 COMMON STATUS 66 bytes Acc72EX[i].Data8[78].a IN (neXto Hots System) DPM (DataPort Memory) UNCONTROLED 0 CXTENDED STATUS 422 bytes Acc72EX[i].Data8[848].a IN (neXto Hots System) DPM (DataPort Memory) DPM (DataPort Memory) DPM (DataPort Memory) DPM (DataPort Memory) DPM (DataPort Memory)	256 bytes Acc722K(I)Data8[512] a Communication 15616 bytes Acc722K(I)Data8[52] a MATER BITS Acc722K(I)Data8[52] a Acc722K(I)Data8[52] a Acc722K(I)Data8[52] a MATER bController 0 9 CONTROL 8 bytes Acc722K(I)Data8[52] a MATER Acc722K(I)Data8[52] a MATER CONTROL 8 bytes Acc72K(I)Data8[52] a DUT (Hota System to netX) DVM (Data97ct Memory) UNCONTROLLED 0 COMMON STATUS 66 bytes Acc72K(I)Data8[78] a N(netXto Hota System) DVM (Data97ct Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc72K(I)Data8[78] a N(netXto Hota System) DVM (Data97ct Memory) DVM (Da	256 bytes Acc728(J] Data8[512] a Communication 15616 bytes Acc728(J] Data8[52] a Acc728(J] Data8[52] a Acc728(J] Data8[52] a Acc728(J] Data8[52] a Acc728(J] Data8[52] a SLAVE 0 9 9 CONTROL 8 bytes Acc728(J] Data8[52] a Acc728(J] Data8[52] a CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 0 COMMON STATUS 64 bytes Acc728(J] Data8[78] a IN (netXto Hock System) 0 COMMON STATUS 64 bytes Acc728(J] Data8[78] a IN (netXto Hock System) 0 COMMON STATUS 64 bytes Acc728(J] Data8[78] a IN (netXto Hock System) 0 COMMON STATUS 64 bytes Acc728(J] Data8[84] a IN (netXto Hock System) 0 COMMON STATUS 64 bytes Acc728(J] Data8[84] a IN (netXto Hock System) 0 CMMON STATUS 64 bytes Acc728(J] Data8[84] a 0 CMMON STATUS 64 bytes Acc728(J] Data8[84] a 10 Acc728(J]	256 bytes Acc722 C(I) Data8[512].a Communication 1556 bytes Acc722 C(I) Data8[52].a IN HANSHKK CHANNEL 16 BITS Acc728 (I) Data8[52].a Acc728 (I) Data8[52].a SCANNER b Obvice 0 9 CONTROL 8 bytes Acc722 (I) Data8[76].a OUT (MoL 8 bytes Acc722 (I) Data8[76].a OUT (MoL 9 CONTROL 8 bytes Acc722 (I) Data8[76].a OUT (MoL 9 CONTROL 10 bytes Acc72 (I) Data8[76].a OUT (MoL 10 bytes Acc72 (I) Data8[76].a OUT (MoL 10 bytes Acc72 (I) Data8[76].a OUT (MoL 10 bytes Acc72 (I) Data8[76].a IN (NetX to Hoc System to netX) DPM (Data14 Drt Memory) UNCONTROLED 0 CONMON STATUS 64 bytes Acc72 (I) Data8[76].a IN (netX to Hoc System) DPM (Data14 Drt Memory) UNCONTROLED 0 EXTENDED STATUS 422 bytes Acc72 (I) Data8[76].a IN (netX to Hoc System) DPM (Data14 Drt Memory) DPM (Data14 Drt Memory) DPM (Data74 Drt Memory) DPM (Data74 Drt Memory) DPM (Data74 Drt Memory) DPM (Data74 Drt Memory)	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Downer 0 CONTROL 8 bytes Acc722K[I].Data8[768].a DUT (IndS System to netX) DPM (Dual+Ort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a N (netX to Hox System) DPM (Dual+Dort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I].Data8[784].a N (netX to Hox System) DPM (Dual+Dort Memory) UNCONTROLLED 0 CXTENDED STATUS 432 bytes Acc722K[I].Data8[548].a N (netX to Hox System) DPM (Dual+Dort Memory) DPM (Dual+Dort Memory)	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[512].a Acc72EX[i].Data8[50].a Acc72EX[i].Data8[50].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a CONTROL 8 bytes Acc72EX[i].Data8[76].a OUT (Hots System to netX) DPM (Data9crt Memory) UNCONTROLLED 0 COMMON STATUS 66 bytes Acc72EX[i].Data8[78].a IN (netX to Host System) DPM (Data9crt Memory) UNCONTROLLED 0 EXTENDED STATUS 422 bytes Acc72EX[i].Data8[24].a IN (netX to Host System) DPM (Data9crt Memory) UNCONTROLLED 0	2 55 bytes Acc722K[I],Data8[52],J Communication 1 5516 bytes Acc722K[I],Data8[520],J NH HANDSHAKC CHANNEL 16 BITS Acc722K[I],Data8[520],J Acc722K[I],Data8[52],J IO-CONTROLES Programmable Logic Controller (PIc) 0 9 CONTROL 8 bytes Acc722K[I],Data8[768],J OUT (Intex5 System to netX) DWI (Du4Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K[I],Data8[764],J NICREX ID 405 System) DVI (Intex Lin Host System)	256 bytes Acc72EX[i].Data8[512].a Communication 15616 bytes Acc72EX[i].Data8[768].a IN HANDSHAKE GHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT [Host System to netX] DFM (DuaH-Out Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[784].a IN (netX to Host System) DFM (DuaH-Ort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[784].a IN (netX to Host System) DFM (DuaH-Ort Memory) UNCONTROLED 0 EXTENDED STATUS 432 bytes Acc72EX[i].Data8[848].a IN (netX to Host System)
Channel Type Size of Channel Channel Start Address Block 2 Channel Start Address Channel Start Address Position of Handshake Cells Size of Anadhake Cells Size of Handshake Cells NetX Handshake Register Host Handshake Register Conformance Class Number of Subblocks 	256 bytes Acc72EX[i].Data8[512] a Communication 15516 bytes Acc72EX[i].Data8[576] a Acc72EX[i].Data8[520] a Acc72EX[i].Data8[520] a Acc72EX[i].Data8[520] a Acc72EX[i].Data8[520] a MATER Managing Node 0 9 CONTROL 8 bytes Acc72EX[i].Data8[768] a OUT (Hot System to netX) DFM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[768] a M (netX to Hot System to netX) DFM (Dual-Port Memory) UNCONTROLED 0 EXTENDED STATUS 432 bytes Acc72EX[i].Data8[3,a] N (netX to Hot System) DFM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72EX[i].Data8[3,a] N (netX to Hot System) DFM (Dual-Port Memory) UNCONTROLED 0 MALBOX 1600 bytes Acc72EX[i].Data8[1280] a	256 bytes Acc722K[I].Data8[512].a Communication 7424 bytes Acc722K[I].Data8[522].a Kac722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes Acc722K[I].Data8[768].a OUT (Hots System to netX) DPM (Data910 ort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes COMMON STATUS 64 bytes CO	255 bytes Acc722C(I)[Data8[512] a Communication 1556 bytes Acc722C(I)[Data8[522] a MAKE CHANNEL 16 BITS Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a Acc722C(I)[Data8[520] a CONTROL 8 bytes Acc722C(I)[Data8[520] a OUT (Hot System to netX) DPM (Dual-ArX Memory) UNCONTROLED 0 COMMON STATUS 6 bytes Acc722(I)[Data8[548] a N(netX to Hot System) DPM (Dual-ArX Memory) UNCONTROLED 0 COMMON STATUS 422 bytes Acc722(I)[Data8[548] a N(netX to Hot System) DPM (Dual-ArX Memory) UNCONTROLED 0 CXENDED STATUS 422 bytes Acc722(I)[Data8[548] a N(netX to Hot System) DPM (Dual-ArX Memory) UNCONTROLED 0 MALBOX 1000 bytes Acc722(I)[Data8[548] a N(netX to Hot System) DPM (Dual-ArX Memory) UNCONTROLED 0 MALBOX	256 bytes Acc728/[i]Data8[512] a Communication 7424 bytes Acc728/[i]Data8[582],a Acc728/[i]Data8[580],a Acc728/[i]Data8[530],a Acc728/[i]Data8[530],a Acc728/[i]Data8[530],a Acc728/[i]Data8[530],a Acc728/[i]Data8[522],a Server 0 9 CONTROL 8 bytes Acc728/[i]Data8[52],a Acc728/[i]Data8[52],a Acc728/[i]Data8[52],a CONTROL 8 bytes CONTROL 8 bytes COMMON 513/TUS 6 bytes Acc728/[i]Data8[784],a MinexXio Host System to netX) DPM (Dua14Port Memory) UNCONTROLLED 0 COMMON 513/TUS 422 bytes Acc728/[i]Data8[84],a Ni (netX to Host System) DPM (Dua14Port Memory) UNCONTROLLED 0 MALIBOX 1600 bytes Acc728/[i]Data8[84],a Ni (netX to Host System) DPM (Dua14Port Memory) UNCONTROLLED 0	255 bytes Acc722K[i].bata8[52].a Communication 15516 bytes Acc722K[i].bata8[576].a IN HANDENAKE CHANNEL 16 BITS Acc722K[i].bata8[52].a Acc722K[i].bata8[52].a MASTER Scanner 0 9 CONTROL 8 bytes Acc722K[i].bata8[768].a OUT (Hots System to netX) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 6 dbytes Acc72K[i].bata8[78].a MASTER Master System to netX) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 432 bytes Acc72K[i].bata8[78].a Micret Kin Hots System) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 432 bytes Acc72K[i].bata8[78].a IN (netX to Hots System) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 432 bytes Acc72K[i].bata8[78].a IN (netX to Hots System) DPM (Dual-Port Memory) UNCONTROLED 0 MALBOX 1600 bytes Acc72K[i].bata8[220].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[512].a IN HAND5HAKC CHANNEL 16 BTIS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a CONTROL 8 bytes CONTROL 8 bytes Acc72EX[i].Data8[768].a OUT (Hock System to netX) DPM (Dual+Ort Memory) UNCONTROLLED 0 COMMON STATUS 6472EX[i].Data8[784].a McC72EX[i].Data8[784].a McC72EX[i].Data8[784].a McC72EX[i].Data8[784].a McC72EX[i].Data8[784].a McC72EX[i].Data8[784].a McC72EX[i].Data8[848].a N(netX to Hock System) DPM (Dual+Ort Memory) UNCONTROLLED 0 CMMON STATUS 432 bytes Acc72EX[i].Data8[848].a N(netX to Hock System) DPM (Dual+Ort Memory) UNCONTROLLED 0 MALBOX 1600 bytes Acc72EX[i].Data8[1280].a	256 bytes Acc72EX[i].Data8[512].a Communication 7424 bytes Acc72EX[i].Data8[512].a IN HANGKAKE CHANNEL 16 BITS Acc72EX[i].Data8[520].a Acc72EX[i].Data8[520].a Acc72EX[i].Data8[522].a SLAVE 0 9 CONTROL 8 bytes Acc72EX[i].Data8[522].a SLAVE 0 9 CONTROL 8 bytes Acc72EX[i].Data8[52].a 0 0 CONTROL 8 bytes Acc72EX[i].Data8[52].a 0 0 CONTROL 8 bytes Acc72EX[i].Data8[52].a 0 0 CONTROL 8 bytes Acc72EX[i].Data8[52].a 0 0 CONTROL 8 bytes Acc72EX[i].Data8[52].a 0 CONTROL 8 bytes Acc72EX[i].Data8[52].a 0 CONTROLED CONTROLED 0 CONTROL	256 bytes Ac;722K(I)Data8[512] a Communication 15616 bytes Ac;722K(I)Data8[520] a Ac;722K(I)Data8[520] a Ac;722K(I)Data8[520] a Ac;722K(I)Data8[520] a Ac;722K(I)Data8[520] a Ac;722K(I)Data8[520] a Ac;722K(I)Data8[520] a MASTER b Controller 0 9 CONTROL 8 bytes Ac;722K(I)Data8[52] a DUT (Hot System to netX) DPM (Data8[784] a Nu KCONTROLED 0 COMMON STATUS 6 bytes Ac;722K(I)Data8[784] a Nu (Ac) Ac)Data8[784] a Nu (Ac)Data8[784] a Nu (Ac)Data8[7	256 bytes Acc722K[I]Data[512]a Communication 1556 bytes Acc722K[I]Data[512]a IN HANDSHAC CHANNEL 16 BITS Acc722K[I]Data[520]a Acc722K[I]Data[520]a Acc722K[I]Data[520]a Acc722K[I]Data[52]a SLAVE 0 9 CONTROL 8 bytes Acc722K[I]Data[76]a OUT (Hot 35 system to netX) DPM (Data]47cr Memory) UNCONTROLED 0 COMMON STATUS 6 bytes Acc722K[I]Data[78]a N(FietXL0 bids System) DPM (Data]47cr Memory) UNCONTROLED 0 COMMON STATUS 6 bytes Acc722K[I]Data[8]48]a N(FietXL0 bids System) DPM (Data]47cr Memory) UNCONTROLED 0 0 MALIBOX 1600 bytes Acc728K[I]Data[8]48]a N(FietXL0 bids System) DPM (Data]47cr Memory) UNCONTROLED 0 MALIBOX	256 bytes Acc722 C(I) Data[512] a Communication 1556 bytes Acc722 C(I) Data[512] a IN HANDSHAC CHANNEL 16 BITS Acc722 C(I) Data[520] a Acc722 C(I) Data[520] a Acc722 C(I) Data[520] a Acc722 C(I) Data[520] a CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 CONT	256 bytes Acc722K[I].Data8[512].a Communication 15616 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a CONTROL 8 bytes Acc722K[I].Data8[768].a DUT [Hot5 System to netX) DPM (Dual-Port Memory) UNCONTROLLED 0 COMMON 5TATUS 432 bytes Acc722K[I].Data8[548].a N (netX to Hot5 System) DPM (Dual-Port Memory) UNCONTROLLED 0 COMMON 5TATUS 432 bytes Acc722K[I].Data8[548].a N (netX to Hot5 System) DPM (Dual-Port Memory) UNCONTROLLED 0 MALIBOX 1600 bytes Acc722K[I].Data8[548].a N (netX to Hot5 System) DPM (Dual-Port Memory) UNCONTROLLED 0	256 bytes Acr22EX[i].Data8[512].a Communication 15516 bytes Acr22EX[i].Data8[512].a IN HANGKAC CHANNEL 16 BITS Acr22EX[i].Data8[520].a Acr22EX[i].Data8[520].a Acr22EX[i].Data8[520].a Acr22EX[i].Data8[520].a Acr22EX[i].Data8[520].a Acr22EX[i].Data8[520].a Acr22EX[i].Data8[520].a Acr22EX[i].Data8[520].a CONTROL 8 bytes Acr22EX[i].Data8[520].a CONTROL 8 bytes Acr22EX[i].Data8[520].a CONTROL 8 bytes Acr22EX[i].Data8[520].a CONTROL 8 bytes Acr22EX[i].Data8[520].a CONTROL 8 bytes Acr22EX[i].Data8[520].a EXTENDED STATUS 422 bytes Acr22EX[i].Data8[54].a IN (neXt of Host System) DFM (Data9ch Memory) UNCONTROLLED 0 MALIBOX 1000 bytes Acr22EX[i].Data8[54].a IN (neXt of Host System) DFM (Data9ch Memory) UNCONTROLLED 0 MALIBOX 1000 bytes Acr22EX[i].Data8[520].a	2 55 bytes Acc722K[I,Data8[52].a Communication 1 5516 bytes Acc722K[I,Data8[520].a Acc722K[I,Data8[520].a Acc722K[I,Data8[520].a Acc722K[I,Data8[52].a IO-CONTROLES Programmable Logic Controller (PIc) 0 9 CONTROL 8 bytes Acc722K[I,Data8[768].a OUT (Hots System to netK) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 6 kytes Acc722K[I,Data8[764].a MI, netK to Hot System) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 432 bytes Acc722K[I,Data8[764].a MI, netK to Hot System) DPM (Dual-Port Memory) UNCONTROLED 0 COMMON STATUS 432 bytes Acc72K[I,Data8[84].a IN (netK to Hot System) DPM (Dual-Port Memory) UNCONTROLED 0 MALBOX 1600 bytes Acc72K[I,Data8[240].a MALBOX 1600 bytes	256 bytes Acc72EX[i]Data8[512]a Communication 15616 bytes Acc72EX[i]Data8[768]a IN HANDSHAC CHANNEL 16 BITS Acc72EX[i]Data8[520]a Acc72EX[i]Data8[520]a Acc72EX[i]Data8[520]a Acc72EX[i]Data8[520]a Acc72EX[i]Data8[520]a CONTROL 8 bytes Acc72EX[i]Data8[768]a OUT (Hot5Yetten to netX) OUT (Hot5Yetten to netX) OVECONTROLED 0 COMMON STATUS EXTENDED STATUS 432 bytes Acc72EX[i]Data8[348]a N (netX to Hot5 System) DPM (Dua14Port Memory) UNCONTROLED 0 MAILBOX 100 bytes Acc72EX[i]Data8[34]a
Channel Type Size of Channel Channel Start Address Size of Channel Channel Start Address Channel Start Address Channel Start Address Contom Start Address Size of Handshake Cells Size of Handshake Cells Size of Handshake Cells NetX Handshake Register Communication Class Protocol Class Conformance Class Number of Subblocks Size Size Offact Transfer Direction Transfer Direction Transfer Direction Transfer Direction Transfer Type Handshake INde Handshake INde Handshake Mode Handshake Mode Handshake Mode Handshake Mode Handshake Mode Handshake Mode Handshake Mode Handshake Mode Handshake Bit Size Size Size Size Contection Transfer Type Handshake Mode Handshake Mode Handshake Bit Size	255 bytes Acc72EX[I].Data8[512].a Communication 1556 bytes Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a Acc72EX[I].Data8[520].a B B COMTPOL B B COMTPOL B D COMTPOL B COMTON COMTPOLED D COMMON STATUS 64 bytes Acc72EX[I].Data8[78].a DIV (IndeJ-Port Memory) UNCONTPOLED 0 COMMON STATUS 64 bytes Acc72EX[I].Data8[78].a N(IndEX to Host System) DFM (IndeJ-Port Memory) UNCONTPOLED 0 COMMON STATUS 64 bytes Acc72EX[I].Data8[78].a N(IndEX to Host System) DFM (IndeJ-Port Memory) UNCONTPOLED 0 Acc72EX[I].Data8[78].a N(IndEX to Host System) DFM (IndeJ-Port Memory) UNCONTPOLED 0 MALBOX 1600 bytes	256 bytes Acc722K[I].Data8[512].a Communication 7424 bytes Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a IN HANDSHAKE CHANNEL 16 BTS Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc722K[I].Data8[520].a Acc72K[I].Data8[520].a COMTROL 8 bytes Acc72K[I].Data8[768].a OUTIFIOL 8 bytes Acc72K[I].Data8[768].a OUT(Host System to netX) DPM (DuaFort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc72K[I].Data8[784].a N (netX to Host System) DPM (DuaFort Memory) UNCONTROLED 0 EXTENDED STATUS 432 bytes Acc72ZK[I].Data8[84].a N (netX to Host System) DPM (DuaFort Memory) UNCONTROLED 0 EXTENDED STATUS 432 bytes Acc72X[I].Data8[84].a N (netX to Host System) DPM (DuaFort Memory) UNCONTROLED 0 M (IABEDX a) Acc72X[I].Dat	255 bytes Acc7225(J],Data8[512],a Communication 1556 bytes Acc7225(J],Data8[512],a MASTER Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a MASTER Bytes Acc7225(J],Data8[768],a OUT (Host System to netX) DPM (Dua1470t Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722D(J],Data8[784],a NI (netX to Not System) DPM (Dua1470t Memory) UNCONTROLED 0 EXTMODE STATUS 42 bytes Acc7225(J],Data8[784],a NI (netX to Not System) DPM (Dua1470t Memory) UNCONTROLED 0 MALBOX 1000 bytes	256 bytes Acc7225(J],Data8[512],a Communication 7424 bytes Acc7225(J],Data8[512],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a Acc7225(J],Data8[520],a CONTROL 8 bytes Acc7225(J],Data8[520],a CONTROL 8 bytes Acc7225(J],Data8[528],a OUT (Host System to netX) DPM (Dua14Port Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc7225(J],Data8[784],a N (netX to Acs System) DPM (Dua14Port Memory) UNCONTROLLED 0 EXTENDED STATUS 427 bytes Acc7225(J],Data8[848],a N (netX to Acs System) DPM (Dua14Port Memory) UNCONTROLLED 0 MULBOX 1600 bytes	255 bytes Acc72EX[i].bata8[512].a Communication 15516 bytes Acc72EX[i].bata8[576].a IN HANDENAKE CHANNEL 16 BITS Acc72EX[i].bata8[52].a Acc72EX[i].bata8[52].a Acc72EX[i].bata8[52].a MASTER Scanner 0 9 9 9 9 9 0 9 0 0 9 0 0 9 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0	256 bytes Acc72EX[I],Data8[512],a Communication 7424 bytes Acc72EX[I],Data8[522],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[520],a Acc72EX[I],Data8[768],a OUT (HotS System to netX) DPM (Dua140 rot Memory) UNCONTROLLED 0 COMMON STATUS G6 bytes Acc72EX[I],Data8[784],a N (netX to HotS System) DPM (Dua140 rot Memory) UNCONTROLLED 0 State8[25],A Acc72EX[I],Data8[784],a N (netX to HotS System) DPM (Dua140 rot Memory) UNCONTROLLED 0 MALBOX 1600 bytes	256 bytes Acc725K(I].Data8[512].a Communication 7424 bytes Acc725K(I].Data8[512].a IN HANGKAK CHANNEL 16 BITS Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a Acc725K(I].Data8[520].a CONTROL 8 bytes Acc725K(I].Data8[784].a OUT (Host System to netX) DPM (Dua1-Port Memory) UNCONTROLLED COMMON STATUS 64 bytes Acc725K(I].Data8[784].a IN (netX to Host System) DPM (Dua1-Port Memory) UNCONTROLLED 0 EXTENDED STATUS 42 bytes Acc725K(I].Data8[88].a IN (netX to Host System) DPM (Dua1-Port Memory) UNCONTROLLED 0 MALISOX	256 bytes Ac:72EX[(].Data8[512].a Communication 156 Ib bytes Ac:72EX[(].Data8[512].a IN HANGENKE CHANNEL Is BITS Ac:72EX[(].Data8[520].a Ac:72EX[(].Data8[520].a Ac:72EX[(].Data8[520].a Ac:72EX[(].Data8[520].a MASTEM b Controller 0 9 9 9 0 0 0 0 0 0 0 0 0 0 0 0 0	256 bytes Acc722K[I]Data8[512].a Communication 15516 bytes Acc722K[I]Data8[52].a Acc722K[I]Data8[52].a SLVE b-Controller 0 CONTROL 8 bytes CONTROL 8 bytes CONTROL 9 9 9 9 CONTROL 8 bytes Acc72X[I]Data8[76].a OUT (Host System to netX) DPM (Dua14Port Memory) UNCONTROLED 0 0 COMMON STATUS 64 bytes Acc722K[I]Data8[78].a IN (InteX to Host System) DPM (Dua14Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K[I]Data8[78].a IN (InteX to Host System) DPM (Dua14Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K[I]Data8[78].a IN (InteX to Host System) DPM (Dua14Port Memory) UNCONTROLED 0 MALBOX 1600 bytes	256 bytes Acc722 Cf(I)Data8[512].a Communication 15616 bytes Acc722 Rf(I)Data8[512].a KAC722 Rf(I)Data8[520].a Acc722 Rf(I)Data8[520].a Acc722 Rf(I)Data8[520].a Acc722 Rf(I)Data8[520].a Acc72 Rf(I)Data8[520].a CONTROL Bytes Acc72 Rf(I)Data8[768].a OUT (Host System to netX) DPM (Dua1-Fort Memory) UNCONTROLED O COMMON STATUS E4 bytes Acc72 Rf(I)Data8[784].a N (netX to Not System) DPM (Dua1-Fort Memory) UNCONTROLED O CONTROLED CONTROLE	256 bytes Acc7225(I).Data8[512].a Communication 15616 bytes Acc7225(I).Data8[520].a Acc7225(I).Data8[520].a Acc7225(I).Data8[520].a Acc7225(I).Data8[520].a Acc7225(I).Data8[520].a Acc7225(I).Data8[520].a Acc725(I).Data8[520].a CONTROL B bytes Acc725(I).Data8[768].a OUT (Host System to netX) DPM (Dua14Fort Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc7252(I).Data8[784].a N (netX to Host System) DPM (Dua14Fort Memory) UNCONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED 0 CONTROLED 0 MILEOX N (netX to Host System) DPM (Dua14Fort Memory) UNCONTROLED 0 MILEOX N (netX to Host System) DPM (Dua14Fort Memory) UNCONTROLED 0 MILEOX 1600 bytes	256 bytes Acc722K(I)Data8(512).a Communication 15616 bytes Acc722K(I)Data8(522).a MRLAKC CHANNEL 16 BITS Acc722K(I)Data8(520).a Acc722K(I)Data8(522).a MRLSAKGN Combination Firmware 0 9 CONTROL 8 bytes Acc722K(I)Data8(788).a DUT (Host System to netX) DPM (DuaH-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K(I)Data8(784).a IN (netX to Host System) DPM (DuaH-Port Memory) UNCONTROLED 0 COMMON STATUS 64 bytes Acc722K(I)Data8(84).a IN (netX to Host System) DPM (DuaH-Port Memory) UNCONTROLED 0 EXTENDED STATUS 42 bytes Acc722K(I)Data8(84).a IN (netX to Host System) DPM (DuaH-Port Memory) UNCONTROLED 0 MALBOX	2 55 bytes Acc722K[I],Data8[522],a Communication 1 5516 bytes Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[520],a Acc722K[I],Data8[522],a Acc722K[I],Data8[522],a Acc722K[I],Data8[522],a Do-CONTROL Bytes Programmable Logic Controller (PIc) 0 9 9 9 9 COMMON STATUS 64 bytes Acc722K[I],Data8[768],a OUT (Hots System to netX) DPM (Dua1+Ort Memory) UNCONTROLLED 0 COMMON STATUS 64 bytes Acc722K[I],Data8[784],a NI (netX to Hot System) DPM (Dua1+Ort Memory) UNCONTROLLED 0 EXTENDED STATUS 432 bytes Acc722K[I],Data8[784],a NI (netX to Hot System) DPM (Dua1+Ort Memory) UNCONTROLLED 0 MALBOX 1600 bytes	256 bytes Acc72EX[1].Data8[512].a Communication 15616 bytes Acc72EX[1].Data8[52].a IN HANDSHAKE GHANNEL 16 BITS Acc72EX[1].Data8[52].a Acc72EX[1].Data8[52].a Acc72EX[1].Data8[52].a Acc72EX[1].Data8[52].a Acc72EX[1].Data8[52].a Acc72EX[1].Data8[52].a Acc72EX[1].Data8[58].a OUT (Host System to netX) DPM (Dua1Port Memory) UNCONTROLLED 0 COMMON STATUS 66 bytes Acc72EX[1].Data8[78].a IN (netX to Host System) DPM (Dua1Port Memory) UNCONTROLLED 0 STTENDED STATUS 64 bytes Acc72EX[1].Data8[88].a IN (netX to Host System) DPM (Dua1Port Memory) UNCONTROLLED 0 MAILBOX 1600 bytes

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP Scanner/Master	EtherNet/IP Adapter/Slave	Open Modbus/TCP	PROFINET IO Controller/Master	PROFINET IO Device/Slave
Subblock 4	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX	MAILBOX
Size Start Offset	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a	1600 bytes Acc72EX[i].Data8[2280].a
Transfer Direction Transfer Type	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)
Handshake Mode Handshake Bit	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5	UNKNOWN 5
ubblock 5	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE
Size	5760 bytes	1536 bytes	5760 bytes	1536 bytes	5760 bytes	1536 bytes	1536 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes	5760 bytes
Start Offset Transfer Direction	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)	Acc72EX[i].Data8[4480].a OUT (Host System to netX)
Transfer Type Handshake Mode	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROL
Handshake Bit	6	6	6	6	6	6	6	6	6	6	6	6	6	6
Subblock 6	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE	PROCESS DATA IMAGE
Size Start Offset	5760 bytes Acc72EX[i].Data8[10240].a	1536 bytes Acc72EX[i].Data8[6016].a	5760 bytes Acc72EX[i].Data8[10240].a	1536 bytes Acc72EX[i].Data8[6016].a	5760 bytes Acc72EX[i].Data8[10240].a	1536 bytes Acc72EX[i].Data8[6016].a	1536 bytes Acc72EX[i].Data8[6016].a	5760 bytes Acc72EX[i].Data8[10240].a	5760 bytes Acc72EX[i].Data8[10240].a	5760 bytes Acc72EX[i].Data8[10240].a	5760 bytes Acc72EX[i].Data8[10240].a	5760 bytes Acc72EX[i].Data8[10240].a	5760 bytes Acc72EX[i].Data8[10240].a	5760 bytes Acc72EX[i].Data8[10240].a
Transfer Direction Transfer Type	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)
Handshake Mode Handshake Bit	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROL
	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA MAGE	,
Subblock 7 Size	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	64 bytes	HIGH PRIORITY DATA IMAG 64 bytes
Start Offset Transfer Direction	Acc72EX[i].Data8[16000].a OUT (Host System to netX)	Acc72EX[i].Data8[7552].a OUT (Host System to netX)	Acc72EX[i].Data8[16000].a OUT (Host System to netX)	Acc72EX[i].Data8[7552].a OUT (Host System to netX)	Acc72EX[i].Data8[16000].a OUT (Host System to netX)	Acc72EX[i].Data8[7552].a OUT (Host System to netX)	Acc72EX[i].Data8[7552].a OUT (Host System to netX)	Acc72EX[i].Data8[16000].a OUT (Host System to netX)	Acc72EX[i].Data8[16000].a OUT (Host System to netX)	Acc72EX[i].Data8[16000].a OUT (Host System to netX)	Acc72EX[i].Data8[16000].a OUT (Host System to netX)	Acc72EX[i].Data8[16000].a OUT (Host System to netX)	Acc72EX[i].Data8[16000].a OUT (Host System to netX)	Acc72EX[i].Data8[16000].a OUT (Host System to netX)
Transfer Type Handshake Mode	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED	DPM (Dual-Port Memory) BUFFERED, HOST CONTROL
Handshake Bit	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Subblock 8	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAGE	HIGH PRIORITY DATA IMAG
Size Start Offset	64 bytes Acc72EX[i].Data8[16064].a	64 bytes Acc72EX[i].Data8[7616].a	64 bytes Acc72EX[i].Data8[16064].a	64 bytes Acc72EX[i].Data8[7616].a	64 bytes Acc72 EX[i].Data8[16064].a	64 bytes Acc72EX[i].Data8[7616].a	64 bytes Acc72EX[i].Data8[7616].a	64 bytes Acc72EX[i].Data8[16064].a	64 bytes Acc72EX[i].Data8[16064].a	64 bytes Acc72EX[i].Data8[16064].a	64 bytes Acc72EX[i].Data8[16064].a	64 bytes Acc72EX[i].Data8[16064].a	64 bytes Acc72EX[i].Data8[16064].a	64 bytes Acc72EX[i].Data8[16064].a
Transfer Direction Transfer Type	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System)	IN (netX to Host System) DPM (Dual-Port Memory)	IN (netX to Host System) DPM (Dual-Port Memory)
Handshake Mode Handshake Bit	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROLLED 9	BUFFERED, HOST CONTROL 9
Hock 3														
Channel Type Size of Channel	Undefined 0 bytes	Undefined 0 bytes	Undefined 0 bytes	Undefined 0 bytes	Undefined 0 bytes	Undefined 0 bytes	Undefined 0 bytes	Undefined 0 bytes	Undefined O bytes	Communication 15616 bytes	Undefined 0 bytes	Undefined 0 bytes	Undefined 0 bytes	Communication 15616 bytes
Channel Start Address Position of Handshake Cells	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	Acc72EX[i].Data8[212992].a IN HANDSHAKE CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	NOT AVAILABLE BEGINNING OF CHANNEL	Acc72EX[i].Data8[212992]. IN HANDSHAKE CHANNEL
Size of Handshake Cells	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	16 BITS	NOT AVAILABLE	NOT AVAILABLE	NOT AVAILABLE	16 BITS
letX Handshake Register lost Handshake Register	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	Acc72EX[i].Data8[524].a Acc72EX[i].Data8[526].a	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	NOT AVAILABLE NOT AVAILABLE	Acc72EX[i].Data8[524].a Acc72EX[i].Data8[526].a
Communication Class Protocol Class	UNDEFINED	UNDEFINED UNDEFINED	UNDEFINED UNDEFINED	UNDEFINED UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	MESSAGING UNDEFINED	UNDEFINED	UNDEFINED	UNDEFINED	MESSAGING UNDEFINED
Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	U	U	U	0	0	0	0	U	9	0	0	0	9
Subblock 0 Size										CONTROL 8 bytes				CONTROL 8 bytes
Start Offset Transfer Direction										Acc72EX[i].Data8[213000].a OUT (Host System to netX)				Acc72EX[i].Data8[213000].a OUT (Host System to netX)
Transfer Type										DPM (Dual-Port Memory)				DPM (Dual-Port Memory)
Handshake Mode Handshake Bit										UNCONTROLLED 0				UNCONTROLLED 0
- Subblock 1										COMMON STATUS				COMMON STATUS
Size Start Offset										64 bytes Acc72 EX[i].Data8[213008].a				64 bytes Acc72EX[i].Data8[213008].a
Transfer Direction										IN (netX to Host System)				IN (netX to Host System)
Transfer Type Handshake Mode										DPM (Dual-Port Memory) UNCONTROLLED				DPM (Dual-Port Memory) UNCONTROLLED
Handshake Bit										0				0
- Subblock 2										EXTENDED STATUS				EXTENDED STATUS
Size Start Offset										432 bytes Acc72EX[i].Data8[213072].a				432 bytes Acc72EX[i].Data8[213072].a
Transfer Direction Transfer Type										IN (netX to Host System) DPM (Dual-Port Memory)				IN (netX to Host System) DPM (Dual-Port Memory)
Handshake Mode										UNCONTROLLED				UNCONTROLLED
Handshake Bit										0				0
Subblock 3 Size										MAILBOX 1600 bytes				MAILBOX 1600 bytes
Start Offset										Acc72EX[i].Data8[213504].a				Acc72EX[i].Data8[213504].a
Transfer Direction Transfer Type										OUT (Host System to netX) DPM (Dual-Port Memory)				OUT (Host System to netX) DPM (Dual-Port Memory)
Handshake Mode Handshake Bit										BUFFERED, HOST CONTROLLED 4				BUFFERED, HOST CONTROL 4
										MAUDOX				MAUROX
Subblock 4 Size										MAILBOX 1600 bytes				MAILBOX 1600 bytes
Start Offset Transfer Direction										Acc72EX[i].Data8[215104].a IN (netX to Host System)				Acc72EX[i].Data8[215104]. IN (netX to Host System)
Transfer Type Handshake Mode										DPM (Dual-Port Memory) UNKNOWN				DPM (Dual-Port Memory) UNKNOWN
Handshake Bit										5				5
Subblock 5										PROCESS DATA IMAGE				PROCESS DATA IMAGE
Size Start Offset										5760 bytes Acc72EX[i].Data8[217088].a				5760 bytes Acc72EX[i].Data8[217088].
Transfer Direction Transfer Type										OUT (Host System to netX) DPM (Dual-Port Memory)				OUT (Host System to netX) DPM (Dual-Port Memory)
Handshake Mode										BUFFERED, HOST CONTROLLED				BUFFERED, HOST CONTROL
Handshake Bit										0				6
Subblock 6 Size										PROCESS DATA IMAGE 5760 bytes				PROCESS DATA IMAGE 5760 bytes
Start Offset										Acc72EX[i].Data8[222848].a				Acc72EX[i].Data8[222848].
Transfer Direction Transfer Type										IN (netX to Host System) DPM (Dual-Port Memory)				IN (netX to Host System) DPM (Dual-Port Memory)
Handshake Mode Handshake Bit										BUFFERED, HOST CONTROLLED 7				BUFFERED, HOST CONTROL 7
Subblock 7										HIGH PRIORITY DATA IMAGE				HIGH PRIORITY DATA IMAG
Size										64 bytes				64 bytes
Start Offset Transfer Direction										Acc72EX[i].Data8[216704].a OUT (Host System to netX)				Acc72EX[i].Data8[216704].a OUT (Host System to netX)
Transfer Type Handshake Mode										DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED				DPM (Dual-Port Memory) BUFFERED, HOST CONTROL
Handshake Bit										8				8
Subblock 8										HIGH PRIORITY DATA IMAGE				HIGH PRIORITY DATA IMAG
Size Start Offset										64 bytes Acc72EX[i].Data8[216768].a				64 bytes Acc72EX[i].Data8[216768].
Transfer Direction										IN (netX to Host System)				IN (netX to Host System)
Transfer Type Handshake Mode										DPM (Dual-Port Memory) BUFFERED, HOST CONTROLLED				DPM (Dual-Port Memory) BUFFERED, HOST CONTROL
Handshake Bit										9				9
ock 4 hannel Type	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined	Undefined
e of Channel	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes	0 bytes

	PROFIBUS-DP Master	PROFIBUS-DP Slave	DeviceNet Master	DeviceNet Slave	CANopen Master	CANopen Slave	CC-Link Slave	EtherCAT Master	EtherCAT Slave	EtherNet/IP	EtherNet/IP	Open Modbus/TCP	PROFINET IO	PROFINET IO
										Scanner/Master	Adapter/Slave		Controller/Master	Device/Slave
Channel Start Address	NOT AVAILABLE													
Position of Handshake Cells	BEGINNING OF CHANNEL													
Size of Handshake Cells	NOT AVAILABLE													
NetX Handshake Register	NOT AVAILABLE													
Host Handshake Register	NOT AVAILABLE													
Communication Class	UNDEFINED													
Protocol Class	UNDEFINED													
Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Number of Subblocks	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Block 5														
Channel Type	Undefined													
Size of Channel	0 bytes													
Channel Start Address	NOT AVAILABLE													
Position of Handshake Cells	BEGINNING OF CHANNEL													
Size of Handshake Cells	NOT AVAILABLE													
NetX Handshake Register	NOT AVAILABLE													
Host Handshake Register	NOT AVAILABLE													
Communication Class	UNDEFINED													
Protocol Class	UNDEFINED													
Conformance Class	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Number of Subblocks	0	0	0	0	0	0	0	0	0	0	0	0	0	0