

**SONET/SDH Framer Core
STM0/OC1, STM1/OC3, STM4/OC12
Single and Multi-Channel**

1. Features

The Aliathon STM0/1/4 Framers core provides a flexible, resource-efficient, programmable-logic based solution for SDH interfacing. It caters for both concatenated payloads, such as VC4-4c over STM4, and channelised applications, such as multiple VC3s over STM1. The core...

- Generates SDH frames for STM0, STM1 and STM4. The core may be dynamically switched between SDH rates.
- Scrambles the SDH frame, inserts Regenerator Section Overhead, and calculates B1 values.
- Inserts Multiplex Section Overhead and calculates B2 values.
- Generates all AU pointers (up to 12 for VC3 over STM4).
- Inserts VC3, VC4 and VC4-4c, both channelised and single-channel. All legal combinations of VCs are supported. The VC settings may be dynamically reconfigured, and made on a per VC basis.
- Generates B3 values for all VCs and inserts Higher Order Path Overhead.

2. Functional Description

Figure 2 illustrates the major functional blocks within the SDH Framers Core.

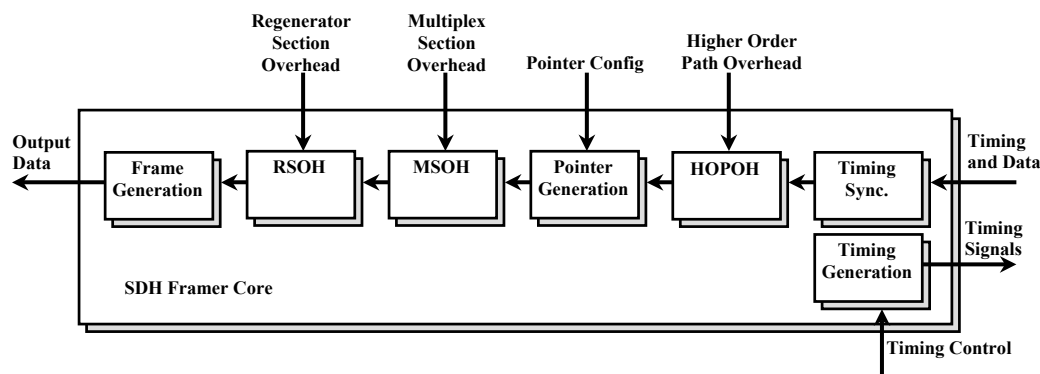


Figure 2 – SDH Deframer Core Functional Blocks

2.1. Frame Generation

The Frame generation block generates the basic SDH frame structure.

2.2. Regenerator Section Overhead

The RSOH block scrambles the outgoing SDH frame (scrambling may be enabled or disabled). It calculates the B1 value and inserts it into the outgoing SDH frame. It also inserts Regenerator Section Overhead supplied by external logic.

2.3. Multiplex Section Overhead

The Multiplex Section Overhead block calculates the B2 value and inserts it into the outgoing frame. It also inserts Multiplex Section Overhead.

2.4. Pointer Generation

The Pointer Generation block calculates all the outgoing AU pointers (up to 12 for VC3 over STM4), and inserts pointer movements.

2.5. Higher Order Path Overhead

The HOPOH block inserts Higher Order Path Overhead, and calculates the B3 value, for all the configured VCs.

2.6. Timing Generation

The Timing Generation block generates a complete SDH frame structure, including VC information. All legal combinations of VCs can be generated. In “simple” configuration mode the core generates one or more of the same VC type (for example, 1 VC4 over STM1, or 12 VC3s over STM4). In “advanced” configuration mode a different VC type can be specified for each potential VC location, allowing for mixes of VC sizes to be extracted. Figure 2 illustrates the SDH multiplex structure and shows how different VC sizes may be combined. Over STM0 VC3 is the only possibility. An STM1 may carry 1 VC4 or 3 VC3s. For STM4 the possibilities are more complex – it may carry 1 VC4-4c, or 4 AUG4s (which can each consist of 3 VC3s or 1 VC4), and thus combinations such as 2 VC4s and 6 VC3s, or 3 VC4s and 3 VC3s are possible. The core can extract all such combinations.

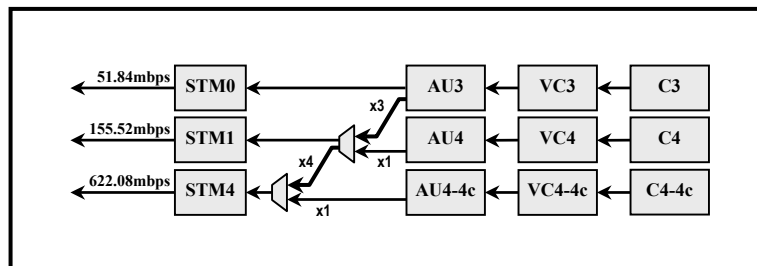


Figure 2 – SDH Multiplex Structure

3. Signal Description

The input and output signals are grouped by function into the following interfaces.

- SDH data output interface
- UpstreamTiming Output interface
- UpstreamTiming/Data Input interface
- Status and configuration interface

3.1. SDH Data Output Interface

This interface provides the input clock and data outputs from the framer core. Typically these signals would drive an external CDR device, but the core also provides an chip-enable signal (**tx_req_vld**) to regulate its output so that it maybe be used to drive other data-sinks (such as asynchronous fifos).

Name	Type	Description
reset	I	Asynchronous reset input.
tx_clk	I	tx_clk is typically supplied by the external LIU. All inputs and outputs from the core are synchronous to this clock unless otherwise noted.
tx_req_vld	I	tx_req_vld is the global chip-enable for the design. It may be used to regulate the amount of data generated by the core for matching with other clock rates. In typical applications, such as when the framer is connected to an external LIU, this input should be tied high.
tx_req_sync	I	tx_req_sync sets the SDH Framer to a known frame position. In typical applications, such as when the framer is connected to an external LIU, this input should be tied low.
tx_vld	O	This output indicates that the data on tx_data is valid. It is essentially a delayed version of the tx_req_vld input. In most applications, such as when the framer is driving an external LIU, this output should be left unconnected.
tx_data (7..0)	O	This is the byte-wide SDH data output. In most applications this will be connected directly to the byte wide transmit interface of an external LIU.
tx_a1_sync	O	When asserted high, this output indicates that the data byte on tx_data is the first SDH A1 framing byte. In most applications this output can be left unconnected.

3.2. Upstream Timing Output

This interface will typically drive an upstream device with SDH and VC timing signals, such as Aliathon's SDH mapper core.

Name	Type	Description
uto_sdh_vld	0	When asserted this indicates that the following outputs are valid.
uto_sdh_sof	0	When asserted this indicates that this clock cycle is the first A1 byte of the SDH frame.
uto_sdh_colcnt (3..0)	0	This output indicates the SDH sub-column valid in this cycle. It will always be 0 for STM0, range from 0 (0000) to 2 (0010) for STM1 and from 0 (0000) to 11 (1011) for STM4.
uto_sdh_toh	0	This output indicates that this clock cycle is fixed transport overhead (ie: Regenerator Section Overhead, Multiplex Section Overhead and AU pointers).
uto_sdh_size (1..0)	0	This output indicates the type of SDH frame being generated. Valid values are... 00 : STM0 01 : STM1 10 : STM4
uto_vc_id (3..0)	0	This output identifies which VC is currently valid. If the input data is all from a single VC (for example, VC3 over STM0 or VC4 over STM1) then this input may be tied to 0 (0000). If the input is from a number of VCs (for example, VC3 or VC4 over STM4) then this input may range from 0 (0000) to 11 (1011).
uto_vc_size (1..0)	0	This output indicates the type of VC that is currently valid. Defined values are... 00 : VC3 01 : VC4 10 : VC4-4c
uto_vc_vld	0	This output indicates that the VC indicated by uto_vc_id is valid. This output is deasserted for transport overhead.
uto_vc_pldvid	0	When asserted this output indicates valid VC payload. This excludes VC Path Overhead and stuff columns.
uto_vc_j1	0	When asserted this indicates that the J1 byte for the VC indicated by uto_vc_id is currently on.
uto_vc_poh	0	When asserted this indicates that the VC Path Overhead for the VC indicated by uto_vc_id is currently valid.
uto_vc_h4cnt (1..0)	0	This counter indicates the H4 multiframe count for structured VC payloads. It ranges between 0 (00) and 3 (11).
uto_vc_ptr (2..0)	0	This output indicates AU pointer actions for the current VC. It is encoded as... 000 : No action 001 : Pointer Increment 010 : Pointer Decrement 100 : New pointer (no NDF) 101 : New pointer (valid NDF)

3.3. Upstream Timing/Data Input

This interface will typically be driven by an upstream device generating SDH and VC timing and data signals, such as Aliathon's SDH mapper core.

Name	Type	Description
uti_sdh_vld	I	When asserted this indicates that the following inputs are valid. In many applications this input can be tied high.
uti_sdh_sof	I	When asserted this indicates that clock cycle is the first A1 byte of the SDH frame. This input is not essential for the operation of the core and may be tied low.
uti_sdh_colcnt (3..0)	I	This input indicates the SDH sub-column valid in this cycle. It will always be 0 for STM0, range from 0 (0000) to 2 (0010) for STM1 and from 0 (0000) to 11 (1011) for STM4.
uti_sdh_toh	I	This input indicates that this clock cycle is fixed transport overhead (ie: Regenerator Section Overhead, Multiplex Section Overhead and AU pointers).
uti_sdh_size (1..0)	I	This input indicates the type of SDH frame being generated. Valid values are... 00 : STM0 01 : STM1 10 : STM4
uti_vc_id	I	This input identifies which VC is currently valid. If the input data is all from a single VC (for example, VC3 over STM0 or VC4 over STM1) then this input may be tied to 0 (0000). If the input is from a number of VCs (for example, VC3 or VC4 over STM4) then this input may range from 0 (0000) to 11 (1011).
uti_vc_size	I	This input indicates the type of VC that is currently valid. Defined values are... 00 : VC3 01 : VC4 10 : VC4-4c
uti_vc_vld	I	This input indicates that the VC indicated by uti_vc_id is valid. This input is deasserted for transport overhead.
uti_vc_pldvld	I	When asserted this input indicates valid VC payload. This excludes VC Path Overhead and stuff columns.
uti_vc_j1	I	When asserted this indicates that the J1 byte for the VC indicated by uti_vc_id is currently valid.
uti_vc_poh	I	When asserted this indicates that the VC Path Overhead for the VC indicated by uti_vc_id is currently valid.
uti_vc_h4cnt (1..0)	I	This counter indicates the H4 multiframe count for structured VC payloads. It ranges between 0 (00) and 3 (11).
uti_vc_ptr (2..0)	I	This input indicates AU pointer actions for the current VC. It is encoded as... 000 : No action 001 : Pointer Increment 010 : Pointer Decrement 100 : New pointer (no NDF) 101 : New pointer (valid NDF)

Name	Type	Description
uti_vc_increq	I	This is an out-of-band request for an AU pointer increment, and should be used for rate-adaptation (for example, if the VC data is provided via an asynchronous fifo that is becoming full). If it is asserted the downstream Timing Generator will cause an AU pointer increment for the current VC (uti_vc_id) in the next frame.
uti_vc_decreq	I	This operates the same as uti_vc_increq but for pointer decrements.
uti_data (7..0)	I	This is the frame data input associated with the above timing inputs.

3.4. Status and Configuration Interface

This interface allows the core to be dynamically configured, and provides status outputs. To lower resource utilisation, any unused outputs should be left open, and configuration inputs should be hard-wired to the required value if they do not need to change. The following configuration and status signals are grouped by function.

3.4.1. General Configuration

Name	Type	Description
gc_rate (1..0)	I	This configuration inputs sets the SDH rate of the core. Defined values are... 00 : STM0 (OC1) 01 : STM1 (OC3) 10 : STM4 (OC12)
gc_sonet	I	When asserted high this puts the core in SONET mode, otherwise it is in SDH mode. The difference between the two is almost non-existent, and does not generally affect inter-operability of equipment. Specifically, this input affects the value of the SS bits in transmitted pointers, and whether VC3 stuff columns are included in B3 calculations or not.

3.4.2. Timing Generation Pointer Control

This interface allows external logic to manipulate the value of pointers, and hence the alignment of VCs within the SDH frame. These inputs connect to the Timing Generation block, and affect the alignment of signals on the Timing Output Interface.

Name	Type	Description
au_ptr_id (3..0)	O	This output indicates what AU the following operations apply to.
au_ptr_vld	O	When asserted high this output indicates the opportunity to introduce a change in pointer for the AU indicated by the au_ptr_id output. The following signals may be asserted in the clock-cycle following to introduce a change in pointer state.
au_ptr_inc	I	When asserted high this causes a pointer increment on the AU indicated by au_ptr_id . This input should be valid the clock-cycle following au_ptr_vld being asserted.
au_ptr_dec	I	When asserted high this causes a pointer decrement on the AU indicated by au_ptr_id . This input should be valid the clock-cycle following au_ptr_vld being asserted.
au_ptr_ndf	I	When asserted high this causes a new pointer value to be used (accompanied by a "New Data Flag" indication) for the AU indicated by au_ptr_id . The new pointer value should be valid on the au_ptr_val input in the same clock cycle as this input is asserted. This input should be valid the clock-cycle following au_ptr_vld being asserted.
au_ptr_new	I	When asserted high this causes a new pointer value to be used (without a "New Data Flag" indication) for the AU indicated by au_ptr_id . The new pointer value should be valid on the au_ptr_val input in the same clock cycle as this input is asserted. This input should be valid the clock-cycle following au_ptr_vld being asserted.
au_ptr_val (9..0)	I	This input carries the new pointer value to be used when au_ptr_ndf or au_ptr_new are asserted.

3.4.3. Timing Generation VC Configuration

This interface provides configuration control over the types of VCs generated by the Timing Generation Block.

Name	Type	Description																
vcfg_id (3..0)	O	This output indicates what VC location is requesting VC configuration information.																
vcfg_size (1..0)	I	<p>This indicates the type of VC to generate. Defined values are...</p> <p>00 : VC3 01 : VC4 10 : VC4-4c</p> <p>It should be valid for the VC location indicated by vcfg_id in the preceding clock cycle. If all VCs have the same setting then this input may be hardwired to a set value. If different settings are to be applied to different VC locations then this input may be driven by the output of a RAM, with vcfg_id as the address input.</p> <p>Assuming that all VCs are configured the same, the core will generate the following number of VCs, depending on the SDH rate.</p> <table border="1"> <thead> <tr> <th></th> <th>VC3</th> <th>VC4</th> <th>VC4-4c</th> </tr> </thead> <tbody> <tr> <td>STM0</td> <td>1</td> <td>3</td> <td>12</td> </tr> <tr> <td>STM1</td> <td></td> <td>1</td> <td>4</td> </tr> <tr> <td>STM4</td> <td></td> <td></td> <td>1</td> </tr> </tbody> </table>		VC3	VC4	VC4-4c	STM0	1	3	12	STM1		1	4	STM4			1
	VC3	VC4	VC4-4c															
STM0	1	3	12															
STM1		1	4															
STM4			1															

3.4.4. Regenerator Section Overhead

This interface provides RSOH configuration options and the mechanism for inserting and manipulating overhead.

Name	Type	Description
rsoh_los	I	When asserted high this input causes the framer core to transmit Loss-of-Signal.
rsoh_scram_off	I	When asserted high the framer transmits the SDH frame unscrambled. For normal operation this input should be low.
rsoh_ins_b1	I	When asserted high the framer inserts the internally generated B1 value. Otherwise, the value provided on the Data Input interface is used. Typically this input should be high.
rsoh_ins_a1a2	I	When asserted high the framer inserts A1 and A2 SDH framing. Otherwise, the values provided on the Data Input interface are used. Typically this input should be high.
rsoh_vld	O	When high this output indicates to external logic that there is the opportunity to influence RSOH payload values, either by injecting or overwriting values. External logic should respond by driving the following inputs in the following clock cycle.
rsoh_id (3..0)	O	This output identifies what kind of RSOH byte that rsoh_vld is valid for. Defined values are... 0000 : A1 0001 : A2 0010 : J0 0011 : B1 0100 : E1 0101 : F1 0110 : D1 0111 : D2 1000 : D3
rsoh_col (3..0)	O	For STM1 and STM4 there is more than one byte for each RSOH field (although typically only the first is of interest). This output indicates which "column" rsoh_vld is for. For STM1 this output ranges from 0 (0000) to 2 (0010), and for STM4 it ranges from 0 (0000) to 11 (1011).
rsoh_first_col	O	This output indicates that rsoh_col is 0. As most applications only use the first RSOH column, rsoh_vld may be left unconnected and this output used to regulate RSOH payload injection.

Name	Type	Description
rsoh_ins	I	When asserted (in response to rsoh_vld) the rsoh_data and rsoh_mask inputs are used for RSOH injection
rsoh_data (7..0)	I	This data byte is qualified with rsoh_mask and transmitted as RSOH overhead if rsoh_ins is asserted.
rsoh_mask (7..0)	I	This mask is qualified with rsoh_data for RSOH insertion. A bit set to 1 causes the corresponding bit in rsoh_data to be sent as overhead, otherwise the data supplied at the Data Input Interface is used.
rsoh_err	I	When asserted (in response to rsoh_vld) the rsoh_err_mask and rsoh_err_val inputs are used inject errors into the RSOH.
rsoh_err_mask (15..0)	I	This input is used to error RSOH bytes (after they have been inserted via the above inputs). Every pair of bits corresponds to a bit of the RSOH byte ((1..0) to bit 0, (3..2) to bit 1, etc), and is encoded as... 00 : No change 01 : Invert RSOH bit 10 : Replace with associated bit from rsoh_err_val . 11 : Undefined
rsoh_err_val (7..0)	I	If the rsoh_err_mask input overwrites bits in the RSOH overhead, the bits used are taken from this input.

3.4.5. Multiplex Section Overhead

This interface provides MSOH configuration options and the mechanism for inserting and manipulating overhead.

Name	Type	Description
msoh_ais	I	When asserted high this input causes the framer core to transmit Multiplex Section Alarm Indication (MS-AIS).
msoh_ins_b2	I	When asserted high the framer inserts the internally generated B2 value. Otherwise, the value provided on the Data Input interface is used. Typically this input should be high.
msoh_vld	O	When high this output indicates to external logic that there is the opportunity to influence MSOH payload values, either by injecting or overwriting values. External logic should respond by driving the following inputs in the following clock cycle.
msoh_id (4..0)	O	<p>This output identifies what kind of MSOH byte that msoh_vld is valid for. Defined values are...</p> <ul style="list-style-type: none"> 00000 : B2 00001 : K1 00010 : K2 00011 : D4 00100 : D5 00101 : D6 00110 : D7 00111 : D8 01000 : D9 01001 : D10 01010 : D11 01011 : D12 01100 : S1 01101 : M1 01110 : E2 01111 : H1 10000 : H2 10001 : H3 <p>Note that the AU pointer bytes (H1,H2,H3) are included here.</p>
msoh_col (3..0)	O	For STM1 and STM4 there is more than one byte for each MSOH field (although typically only the first is of interest). This output indicates which "column" msoh_vld is for. For STM1 this output ranges from 0 (0000) to 2 (0010), and for STM4 it ranges from 0 (0000) to 11 (1011).
msoh_first_col	O	This output indicates that msoh_col is 0. As most applications only use the first MSOH column, msoh_vld may be left unconnected and this output used to regulate MSOH payload injection.

Name	Type	Description
msoh_ins	I	When asserted (in response to msoh_vld) the msoh_data and msoh_mask inputs are used for MSOH injection
msoh_data (7..0)	I	This data byte is qualified with msoh_mask and transmitted as MSOH overhead if msoh_ins is asserted.
msoh_mask (7..0)	I	This mask is qualified with msoh_data for MSOH insertion. A bit set to 1 causes the corresponding bit in msoh_data to be sent as overhead, otherwise the data supplied at the Data Input Interface is used.
msoh_err	I	When asserted (in response to msoh_vld) the msoh_err_mask and msoh_err_val inputs are used inject errors into the MSOH.
msoh_err_mask (15..0)	I	This input is used to error MSOH bytes (after they have been inserted via the above inputs). Every pair of bits corresponds to a bit of the MSOH byte ((1..0) to bit 0, (3..2) to bit 1, etc), and is encoded as... 00 : No change 01 : Invert MSOH bit 10 : Replace with associated bit from msoh_err_val . 11 : Undefined
msoh_err_val (7..0)	I	If the msoh_err_mask input overwrites bits in the MSOH overhead, the bits used are taken from this input.

3.4.6. AU Pointers

This interface provides control over AU pointer generation.

Name	Type	Description
au_vld	O	When asserted high this output indicates the opportunity for external logic to apply AU pointer configuration to the Au indicated by au_id . The following configuration inputs should be valid the clock cycle following this output being asserted.
au_id (3..0)	O	This output indicates which AU pointer the following configuration inputs will be applied to.
au_ins_ptr	I	When asserted high valid AU pointer bytes are asserted for the current AU. Otherwise the value provided at the Data Input Interface is used.
au_ais	I	When asserted high AU AIS is transmitted for the current AU.

3.4.7. Higher Order Path Overhead

Name	Type	Description
hpoh_vld	O	When this output is asserted external logic may influence VC configuration and overhead, for the VC indicated by hpoh_vc_id . The following inputs should be valid in the clock cycle following this output being asserted.
hpoh_vc_id (3..0)	O	As the core can process up to 12 different VCs, there are up to 12 different Higher Order Path Overheads. This output indicates which of the VCs the following input will be applied to. If the core is processing a single VC (VC4 over STM1 for example) then this output will always be 0 (0000). If it is processing 12 VCs (VC3 over STM4) then it range between 0 (0000) and 11 (1011).
hpoh_id (3..0)	O	This output identifies what kind of HPOH byte hpoh_vld is valid for. Defined values are... 0000 : J1 0001 : B3 0010 : C2 0011 : G1 0100 : F2 0101 : H4 0110 : F3 0111 : K3 1000 : N1
hpoh_ins_b3	I	When asserted high the framer inserts the internally generated B3 value for the current VC. Otherwise, the value provided on the Data Input interface is used. Typically this input should be high.
hpoh_ins_h4	I	When asserted high the framer inserts the VC multiframe count (as provided at the Data Input Interface) into the H4 byte location. Otherwise, the count is not asserted. This input should be high for structured VC payloads.

Name	Type	Description
hpoh_ins	I	When asserted (in response to hpoh_vld) the hpoh_data and hpoh_mask inputs are used for HPOH injection
hpoh_data (7..0)	I	This data byte is qualified with hpoh_mask and transmitted as HPOH overhead if hpoh_ins is asserted.
hpoh_mask (7..0)	I	This mask is qualified with hpoh_data for HPOH insertion. A bit set to 1 causes the corresponding bit in hpoh_data to be sent as overhead, otherwise the data supplied at the Data Input Interface is used.
hpoh_err	I	When asserted (in response to hpoh_vld) the hpoh_err_mask and hpoh_err_val inputs are used inject errors into the HPOH.
hpoh_err_mask (15..0)	I	This input is used to error HPOH bytes (after they have been inserted via the above inputs). Every pair of bits corresponds to a bit of the HPOH byte ((1..0) to bit 0, (3..2) to bit 1, etc), and is encoded as... 00 : No change 01 : Invert HPOH bit 10 : Replace with associated bit from hpoh_err_val . 11 : Undefined
hpoh_err_val (7..0)	I	If the hpoh_err_mask input overwrites bits in the HPOH overhead, the bits used are taken from this input.

4. Implementation Details

4.1. Resource Utilisation

The following figures are calculated assuming that all core IOs are routed off-chip. This results in a worst-case resource utilisation figure, and for any given application the resource utilisation is likely to be lower. The example parts are the slowest (and hence cheapest) offered speed-grade, and the core exceeds performance requirements (77.76MHz for STM4) in these devices.

	Stratix Family Eg : EP1S10F484C7			Stratix GX Family Eg : EP1SGX10CF672C7			Cyclone Family Eg : EP1C6F256C8		
	Used by Core	In example Part	Percentage used	Used by Core	In example Part	Percentage used	Used by Core	In example Part	Percentage used
Logic Elements (LEs)	1065	10570	10%	1065	10570	10%	1053	5980	18%
M512 RAM Blocks	7	94	7%	7	94	7%		n/a	
M4k RAM Blocks	2	60	3%	2	60	3%	9	20	45%
M-RAM Blocks	0	1	0%	0	1	0%		n/a	
Fmax	100MHz			100MHz			90MHz		

	Stratix2 Family Eg : EP2S15F484C5		
	Used by Core	In example Part	Percentage used
ALMs	549	6240	9%
M512 RAM Blocks	6	104	5%
M4k RAM Blocks	2	78	2%
M-RAM Blocks	0	1	0%
Fmax	100MHz		



4.2. Ordering Information

For technical enquiries and ordering please contact Aliathon Ltd at

enquires@aliathon.com

Aliathon Ltd
Evans Business Centre
Pitreavie Court, Dunfermline
United Kingdom
KY11 5UU

Phone +44 (0)1383 737 736

Fax +44 (0)1383 749 501

<http://www.aliathon.com/>