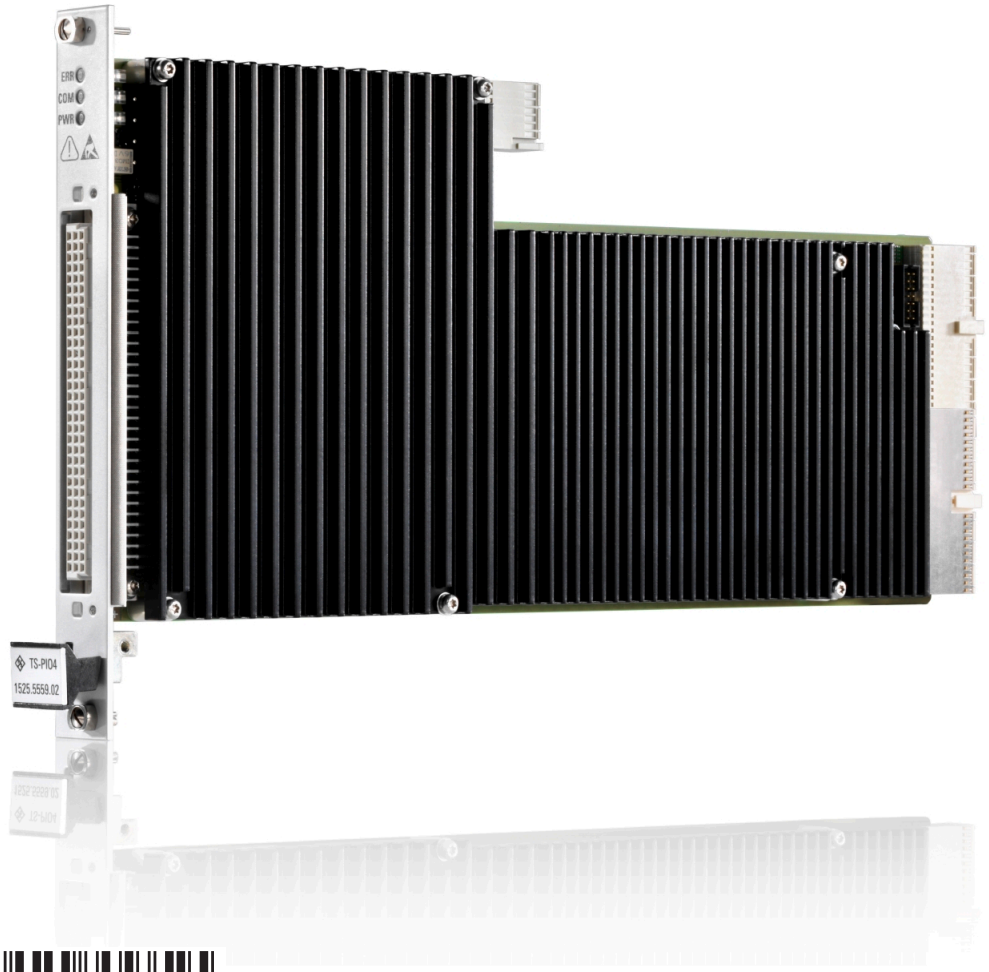
R&S®TS-PIO4
Digital Functional Test Module
User Manual



1178.3192.02 ─ 01

Test & Measurement

User Manual

ROHDE&SCHWARZ

This manual describes the following R&S®TSVP module:

- R&S®TS-PIO4 (1525.5559.02)

Mühldorfstr. 15, 81671 München, Germany
Phone: +49 89 41 29 - 0
Fax: +49 89 41 29 12 164
Email: info@rohde-schwarz.com
Internet: www.rohde-schwarz.com

The following abbreviations are used in this manual: R&S®TS-PIO4 is abbreviated to R&S TS-PIO4.

Basic Safety Instructions

Always read through and comply with the following safety instructions!

All plants and locations of the Rohde & Schwarz group of companies make every effort to keep the safety standards of our products up to date and to offer our customers the highest possible degree of safety. Our products and the auxiliary equipment they require are designed, built and tested in accordance with the safety standards that apply in each case. Compliance with these standards is continuously monitored by our quality assurance system. The product described here has been designed, built and tested in accordance with the EC Certificate of Conformity and has left the manufacturer's plant in a condition fully complying with safety standards. To maintain this condition and to ensure safe operation, you must observe all instructions and warnings provided in this manual. If you have any questions regarding these safety instructions, the Rohde & Schwarz group of companies will be happy to answer them.

Furthermore, it is your responsibility to use the product in an appropriate manner. This product is designed for use solely in industrial and laboratory environments or, if expressly permitted, also in the field and must not be used in any way that may cause personal injury or property damage. You are responsible if the product is used for any purpose other than its designated purpose or in disregard of the manufacturer's instructions. The manufacturer shall assume no responsibility for such use of the product.

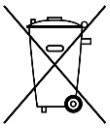
The product is used for its designated purpose if it is used in accordance with its product documentation and within its performance limits (see data sheet, documentation, the following safety instructions). Using the product requires technical skills and, in some cases, a basic knowledge of English. It is therefore essential that only skilled and specialized staff or thoroughly trained personnel with the required skills be allowed to use the product. If personal safety gear is required for using Rohde & Schwarz products, this will be indicated at the appropriate place in the product documentation. Keep the basic safety instructions and the product documentation in a safe place and pass them on to the subsequent users.

Observing the safety instructions will help prevent personal injury or damage of any kind caused by dangerous situations. Therefore, carefully read through and adhere to the following safety instructions before and when using the product. It is also absolutely essential to observe the additional safety instructions on personal safety, for example, that appear in relevant parts of the product documentation. In these safety instructions, the word "product" refers to all merchandise sold and distributed by the Rohde & Schwarz group of companies, including instruments, systems and all accessories. For product-specific information, see the data sheet and the product documentation.

Safety labels on products

The following safety labels are used on products to warn against risks and dangers.

Symbol	Meaning	Symbol	Meaning
	Notice, general danger location Observe product documentation		ON/OFF Power
	Caution when handling heavy equipment		Standby indication
	Danger of electric shock		Direct current (DC)

Symbol	Meaning	Symbol	Meaning
	Caution ! Hot surface		Alternating current (AC)
	Protective conductor terminal To identify any terminal which is intended for connection to an external conductor for protection against electric shock in case of a fault, or the terminal of a protective earth		Direct/alternating current (DC/AC)
	Earth (Ground)		Class II Equipment to identify equipment meeting the safety requirements specified for Class II equipment (device protected by double or reinforced insulation)
	Frame or chassis Ground terminal		EU labeling for batteries and accumulators For additional information, see section "Waste disposal/Environmental protection", item 1.
	Be careful when handling electrostatic sensitive devices		EU labeling for separate collection of electrical and electronic devices For additional information, see section "Waste disposal/Environmental protection", item 2.
	Warning! Laser radiation For additional information, see section "Operation", item 7.		

Signal words and their meaning

The following signal words are used in the product documentation in order to warn the reader about risks and dangers.

DANGER Indicates a hazardous situation which, if not avoided, will result in death or serious injury.

WARNING Indicates a hazardous situation which, if not avoided, could result in death or serious injury.

CAUTION Indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.

NOTICE Indicates information considered important, but not hazard-related, e.g. messages relating to property damage.
In the product documentation, the word ATTENTION is used synonymously.

These signal words are in accordance with the standard definition for civil applications in the European Economic Area. Definitions that deviate from the standard definition may also exist in other economic areas or military applications. It is therefore essential to make sure that the signal words described here are always used only in connection with the related product documentation and the related product. The use of signal words in connection with unrelated products or documentation can result in misinterpretation and in personal injury or material damage.

Operating states and operating positions

The product may be operated only under the operating conditions and in the positions specified by the manufacturer, without the product's ventilation being obstructed. If the manufacturer's specifications are not observed, this can result in electric shock, fire and/or serious personal injury or death. Applicable local or national safety regulations and rules for the prevention of accidents must be observed in all work performed.

1. Unless otherwise specified, the following requirements apply to Rohde & Schwarz products: predefined operating position is always with the housing floor facing down, IP protection 2X, use only indoors, max. operating altitude 2000 m above sea level, max. transport altitude 4500 m above sea level. A tolerance of ±10 % shall apply to the nominal voltage and ±5 % to the nominal frequency, overvoltage category 2, pollution degree 2.
2. Do not place the product on surfaces, vehicles, cabinets or tables that for reasons of weight or stability are unsuitable for this purpose. Always follow the manufacturer's installation instructions when installing the product and fastening it to objects or structures (e.g. walls and shelves). An installation that is not carried out as described in the product documentation could result in personal injury or even death.
3. Do not place the product on heat-generating devices such as radiators or fan heaters. The ambient temperature must not exceed the maximum temperature specified in the product documentation or in the data sheet. Product overheating can cause electric shock, fire and/or serious personal injury or even death.

Electrical safety

If the information on electrical safety is not observed either at all or to the extent necessary, electric shock, fire and/or serious personal injury or death may occur.

1. Prior to switching on the product, always ensure that the nominal voltage setting on the product matches the nominal voltage of the mains-supply network. If a different voltage is to be set, the power fuse of the product may have to be changed accordingly.
2. In the case of products of safety class I with movable power cord and connector, operation is permitted only on sockets with a protective conductor contact and protective conductor.
3. Intentionally breaking the protective conductor either in the feed line or in the product itself is not permitted. Doing so can result in the danger of an electric shock from the product. If extension cords or connector strips are implemented, they must be checked on a regular basis to ensure that they are safe to use.
4. If there is no power switch for disconnecting the product from the mains, or if the power switch is not suitable for this purpose, use the plug of the connecting cable to disconnect the product from the mains. In such cases, always ensure that the power plug is easily reachable and accessible at all times. For example, if the power plug is the disconnecting device, the length of the connecting cable must not exceed 3 m. Functional or electronic switches are not suitable for providing disconnection from the AC supply network. If products without power switches are integrated into racks or systems, the disconnecting device must be provided at the system level.
5. Never use the product if the power cable is damaged. Check the power cables on a regular basis to ensure that they are in proper operating condition. By taking appropriate safety measures and carefully laying the power cable, ensure that the cable cannot be damaged and that no one can be hurt by, for example, tripping over the cable or suffering an electric shock.

6. The product may be operated only from TN/TT supply networks fuse-protected with max. 16 A (higher fuse only after consulting with the Rohde & Schwarz group of companies).
7. Do not insert the plug into sockets that are dusty or dirty. Insert the plug firmly and all the way into the socket provided for this purpose. Otherwise, sparks that result in fire and/or injuries may occur.
8. Do not overload any sockets, extension cords or connector strips; doing so can cause fire or electric shocks.
9. For measurements in circuits with voltages V_{rms} > 30 V, suitable measures (e.g. appropriate measuring equipment, fuse protection, current limiting, electrical separation, insulation) should be taken to avoid any hazards.
10. Ensure that the connections with information technology equipment, e.g. PCs or other industrial computers, comply with the IEC 60950-1 / EN 60950-1 or IEC 61010-1 / EN 61010-1 standards that apply in each case.
11. Unless expressly permitted, never remove the cover or any part of the housing while the product is in operation. Doing so will expose circuits and components and can lead to injuries, fire or damage to the product.
12. If a product is to be permanently installed, the connection between the protective conductor terminal on site and the product's protective conductor must be made first before any other connection is made. The product may be installed and connected only by a licensed electrician.
13. For permanently installed equipment without built-in fuses, circuit breakers or similar protective devices, the supply circuit must be fuse-protected in such a way that anyone who has access to the product, as well as the product itself, is adequately protected from injury or damage.
14. Use suitable overvoltage protection to ensure that no overvoltage (such as that caused by a bolt of lightning) can reach the product. Otherwise, the person operating the product will be exposed to the danger of an electric shock.
15. Any object that is not designed to be placed in the openings of the housing must not be used for this purpose. Doing so can cause short circuits inside the product and/or electric shocks, fire or injuries.
16. Unless specified otherwise, products are not liquid-proof (see also section "Operating states and operating positions", item 1). Therefore, the equipment must be protected against penetration by liquids. If the necessary precautions are not taken, the user may suffer electric shock or the product itself may be damaged, which can also lead to personal injury.
17. Never use the product under conditions in which condensation has formed or can form in or on the product, e.g. if the product has been moved from a cold to a warm environment. Penetration by water increases the risk of electric shock.
18. Prior to cleaning the product, disconnect it completely from the power supply (e.g. AC supply network or battery). Use a soft, non-linting cloth to clean the product. Never use chemical cleaning agents such as alcohol, acetone or diluents for cellulose lacquers.

Operation

1. Operating the products requires special training and intense concentration. Make sure that persons who use the products are physically, mentally and emotionally fit enough to do so; otherwise, injuries or material damage may occur. It is the responsibility of the employer/operator to select suitable personnel for operating the products.

2. Before you move or transport the product, read and observe the section titled "Transport".
3. As with all industrially manufactured goods, the use of substances that induce an allergic reaction (allergens) such as nickel cannot be generally excluded. If you develop an allergic reaction (such as a skin rash, frequent sneezing, red eyes or respiratory difficulties) when using a Rohde & Schwarz product, consult a physician immediately to determine the cause and to prevent health problems or stress.
4. Before you start processing the product mechanically and/or thermally, or before you take it apart, be sure to read and pay special attention to the section titled "Waste disposal/Environmental protection", item 1.
5. Depending on the function, certain products such as RF radio equipment can produce an elevated level of electromagnetic radiation. Considering that unborn babies require increased protection, pregnant women must be protected by appropriate measures. Persons with pacemakers may also be exposed to risks from electromagnetic radiation. The employer/operator must evaluate workplaces where there is a special risk of exposure to radiation and, if necessary, take measures to avert the potential danger.
6. Should a fire occur, the product may release hazardous substances (gases, fluids, etc.) that can cause health problems. Therefore, suitable measures must be taken, e.g. protective masks and protective clothing must be worn.
7. Laser products are given warning labels that are standardized according to their laser class. Lasers can cause biological harm due to the properties of their radiation and due to their extremely concentrated electromagnetic power. If a laser product (e.g. a CD/DVD drive) is integrated into a Rohde & Schwarz product, absolutely no other settings or functions may be used as described in the product documentation. The objective is to prevent personal injury (e.g. due to laser beams).
8. EMC classes (in line with EN 55011/CISPR 11, and analogously with EN 55022/CISPR 22, EN 55032/CISPR 32)
   - Class A equipment:
     Equipment suitable for use in all environments except residential environments and environments that are directly connected to a low-voltage supply network that supplies residential buildings
     Note: Class A equipment is intended for use in an industrial environment. This equipment may cause radio disturbances in residential environments, due to possible conducted as well as radiated disturbances. In this case, the operator may be required to take appropriate measures to eliminate these disturbances.
   - Class B equipment:
     Equipment suitable for use in residential environments and environments that are directly connected to a low-voltage supply network that supplies residential buildings

Repair and service

1. The product may be opened only by authorized, specially trained personnel. Before any work is performed on the product or before the product is opened, it must be disconnected from the AC supply network. Otherwise, personnel will be exposed to the risk of an electric shock.

2. Adjustments, replacement of parts, maintenance and repair may be performed only by electrical experts authorized by Rohde & Schwarz. Only original parts may be used for replacing parts relevant to safety (e.g. power switches, power transformers, fuses). A safety test must always be performed after parts relevant to safety have been replaced (visual inspection, protective conductor test, insulation resistance measurement, leakage current measurement, functional test). This helps ensure the continued safety of the product.

Batteries and rechargeable batteries/cells

If the information regarding batteries and rechargeable batteries/cells is not observed either at all or to the extent necessary, product users may be exposed to the risk of explosions, fire and/or serious personal injury, and, in some cases, death. Batteries and rechargeable batteries with alkaline electrolytes (e.g. lithium cells) must be handled in accordance with the EN 62133 standard.

1. Cells must not be taken apart or crushed.
2. Cells or batteries must not be exposed to heat or fire. Storage in direct sunlight must be avoided. Keep cells and batteries clean and dry. Clean soiled connectors using a dry, clean cloth.
3. Cells or batteries must not be short-circuited. Cells or batteries must not be stored in a box or in a drawer where they can short-circuit each other, or where they can be short-circuited by other conductive materials. Cells and batteries must not be removed from their original packaging until they are ready to be used.
4. Cells and batteries must not be exposed to any mechanical shocks that are stronger than permitted.
5. If a cell develops a leak, the fluid must not be allowed to come into contact with the skin or eyes. If contact occurs, wash the affected area with plenty of water and seek medical aid.
6. Improperly replacing or charging cells or batteries that contain alkaline electrolytes (e.g. lithium cells) can cause explosions. Replace cells or batteries only with the matching Rohde & Schwarz type (see parts list) in order to ensure the safety of the product.
7. Cells and batteries must be recycled and kept separate from residual waste. Rechargeable batteries and normal batteries that contain lead, mercury or cadmium are hazardous waste. Observe the national regulations regarding waste disposal and recycling.

Transport

1. The product may be very heavy. Therefore, the product must be handled with care. In some cases, the user may require a suitable means of lifting or moving the product (e.g. with a lift-truck) to avoid back or other physical injuries.
2. Handles on the products are designed exclusively to enable personnel to transport the product. It is therefore not permissible to use handles to fasten the product to or on transport equipment such as cranes, fork lifts, wagons, etc. The user is responsible for securely fastening the products to or on the means of transport or lifting. Observe the safety regulations of the manufacturer of the means of transport or lifting. Noncompliance can result in personal injury or material damage.
3. If you use the product in a vehicle, it is the sole responsibility of the driver to drive the vehicle safely and properly. The manufacturer assumes no responsibility for accidents or collisions. Never use the product in a moving vehicle if doing so could distract the driver of the vehicle. Adequately secure the product in the vehicle to prevent injuries or other damage in the event of an accident.

Waste disposal/Environmental protection

1. Specially marked equipment has a battery or accumulator that must not be disposed of with unsorted municipal waste, but must be collected separately. It may only be disposed of at a suitable collection point or via a Rohde & Schwarz customer service center.
2. Waste electrical and electronic equipment must not be disposed of with unsorted municipal waste, but must be collected separately.
   Rohde & Schwarz GmbH & Co. KG has developed a disposal concept and takes full responsibility for take-back obligations and disposal obligations for manufacturers within the EU. Contact your Rohde & Schwarz customer service center for environmentally responsible disposal of the product.
3. If products or their components are mechanically and/or thermally processed in a manner that goes beyond their intended use, hazardous substances (heavy-metal dust such as lead, beryllium, nickel) may be released. For this reason, the product may only be disassembled by specially trained personnel. Improper disassembly may be hazardous to your health. National waste disposal regulations must be observed.
4. If handling the product releases hazardous substances or fuels that must be disposed of in a special way, e.g. coolants or engine oils that must be replenished regularly, the safety instructions of the manufacturer of the hazardous substances or fuels and the applicable regional waste disposal regulations must be observed. Also observe the relevant safety instructions in the product documentation. The improper disposal of hazardous substances or fuels can cause health problems and lead to environmental damage.

For additional information about environmental protection, visit the Rohde & Schwarz website.

Instrucciones de seguridad elementales

¡Es imprescindible leer y cumplir las siguientes instrucciones e informaciones de seguridad!

El principio del grupo de empresas Rohde & Schwarz consiste en tener nuestros productos siempre al día con los estándares de seguridad y de ofrecer a nuestros clientes el máximo grado de seguridad. Nuestros productos y todos los equipos adicionales son siempre fabricados y examinados según las normas de seguridad vigentes. Nuestro sistema de garantía de calidad controla constantemente que sean cumplidas estas normas. El presente producto ha sido fabricado y examinado según el certificado de conformidad de la UE y ha salido de nuestra planta en estado impecable según los estándares técnicos de seguridad. Para poder preservar este estado y garantizar un funcionamiento libre de peligros, el usuario deberá atenerse a todas las indicaciones, informaciones de seguridad y notas de alerta. El grupo de empresas Rohde & Schwarz está siempre a su disposición en caso de que tengan preguntas referentes a estas informaciones de seguridad.

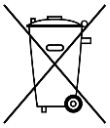
Además queda en la responsabilidad del usuario utilizar el producto en la forma debida. Este producto está destinado exclusivamente al uso en la industria y el laboratorio o, si ha sido expresamente autorizado, para aplicaciones de campo y de ninguna manera deberá ser utilizado de modo que alguna persona/cosa pueda sufrir daño. El uso del producto fuera de sus fines definidos o sin tener en cuenta las instrucciones del fabricante queda en la responsabilidad del usuario. El fabricante no se hace en ninguna forma responsable de consecuencias a causa del mal uso del producto.

Se parte del uso correcto del producto para los fines definidos si el producto es utilizado conforme a las indicaciones de la correspondiente documentación del producto y dentro del margen de rendimiento definido (ver hoja de datos, documentación, informaciones de seguridad que siguen). El uso del producto hace necesarios conocimientos técnicos y ciertos conocimientos del idioma inglés. Por eso se debe tener en cuenta que el producto solo pueda ser operado por personal especializado o personas instruidas en profundidad con las capacidades correspondientes. Si fuera necesaria indumentaria de seguridad para el uso de productos de Rohde & Schwarz, encontraría la información debida en la documentación del producto en el capítulo correspondiente. Guarde bien las informaciones de seguridad elementales, así como la documentación del producto, y entréguelas a usuarios posteriores.

Tener en cuenta las informaciones de seguridad sirve para evitar en lo posible lesiones o daños por peligros de toda clase. Por eso es imprescindible leer detalladamente y comprender por completo las siguientes informaciones de seguridad antes de usar el producto, y respetarlas durante el uso del producto. Deberán tenerse en cuenta todas las demás informaciones de seguridad, como p. ej. las referentes a la protección de personas, que encontrarán en el capítulo correspondiente de la documentación del producto y que también son de obligado cumplimiento. En las presentes informaciones de seguridad se recogen todos los objetos que distribuye el grupo de empresas Rohde & Schwarz bajo la denominación de "producto", entre ellos también aparatos, instalaciones así como toda clase de accesorios. Los datos específicos del producto figuran en la hoja de datos y en la documentación del producto.

Señalización de seguridad de los productos

Las siguientes señales de seguridad se utilizan en los productos para advertir sobre riesgos y peligros.

Símbolo	Significado	Símbolo	Significado
	Aviso: punto de peligro general Observar la documentación del producto		Tensión de alimentación de PUESTA EN MARCHA / PARADA
	Atención en el manejo de dispositivos de peso elevado		Indicación de estado de espera (standby)
	Peligro de choque eléctrico		Corriente continua (DC)
	Advertencia: superficie caliente		Corriente alterna (AC)
	Conexión a conductor de protección		Corriente continua / Corriente alterna (DC/AC)
	Conexión a tierra		El aparato está protegido en su totalidad por un aislamiento doble (reforzado)
	Conexión a masa		Distintivo de la UE para baterías y acumuladores Más información en la sección "Eliminación/protección del medio ambiente", punto 1.

Símbolo	Significado	Símbolo	Significado
	Aviso: Cuidado en el manejo de dispositivos sensibles a la electrostática (ESD)		Distintivo de la UE para la eliminación por separado de dispositivos eléctricos y electrónicos Más información en la sección "Eliminación/protección del medio ambiente", punto 2.
	Advertencia: rayo láser Más información en la sección "Funcionamiento", punto 7.		

Palabras de señal y su significado

En la documentación del producto se utilizan las siguientes palabras de señal con el fin de advertir contra riesgos y peligros.

Indica una situación de peligro que, si no se evita, causa lesiones graves o incluso la muerte.

Indica una situación de peligro que, si no se evita, puede causar lesiones graves o incluso la muerte.

Indica una situación de peligro que, si no se evita, puede causar lesiones leves o moderadas.

Indica información que se considera importante, pero no en relación con situaciones de peligro; p. ej., avisos sobre posibles daños materiales.
En la documentación del producto se emplea de forma sinónima el término CUIDADO.

Las palabras de señal corresponden a la definición habitual para aplicaciones civiles en el área económica europea. Pueden existir definiciones diferentes a esta definición en otras áreas económicas o en aplicaciones militares. Por eso se deberá tener en cuenta que las palabras de señal aquí descritas sean utilizadas siempre solamente en combinación con la correspondiente documentación del producto y solamente en combinación con el producto correspondiente. La utilización de las palabras de señal en combinación con productos o documentaciones que no les correspondan puede llevar a interpretaciones equivocadas y tener por consecuencia daños en personas u objetos.

Estados operativos y posiciones de funcionamiento

El producto solamente debe ser utilizado según lo indicado por el fabricante respecto a los estados operativos y posiciones de funcionamiento sin que se obstruya la ventilación. Si no se siguen las indicaciones del fabricante, pueden producirse choques eléctricos, incendios y/o lesiones graves con posible consecuencia de muerte. En todos los trabajos deberán ser tenidas en cuenta las normas nacionales y locales de seguridad del trabajo y de prevención de accidentes.

1. Si no se convino de otra manera, es para los productos Rohde & Schwarz válido lo que sigue: como posición de funcionamiento se define por principio la posición con el suelo de la caja para abajo, modo de protección IP 2X, uso solamente en estancias interiores, utilización hasta 2000 m sobre el nivel del mar, transporte hasta 4500 m sobre el nivel del mar. Se aplicará una tolerancia de ±10 % sobre el voltaje nominal y de ±5 % sobre la frecuencia nominal. Categoría de sobrecarga eléctrica 2, índice de suciedad 2.
2. No sitúe el producto encima de superficies, vehículos, estantes o mesas, que por sus características de peso o de estabilidad no sean aptos para él. Siga siempre las instrucciones de instalación del fabricante cuando instale y asegure el producto en objetos o estructuras (p. ej. paredes y estantes). Si se realiza la instalación de modo distinto al indicado en la documentación del producto, se pueden causar lesiones o, en determinadas circunstancias, incluso la muerte.
3. No ponga el producto sobre aparatos que generen calor (p. ej. radiadores o calefactores). La temperatura ambiente no debe superar la temperatura máxima especificada en la documentación del producto o en la hoja de datos. En caso de sobrecalentamiento del producto, pueden producirse choques eléctricos, incendios y/o lesiones graves con posible consecuencia de muerte.

Seguridad eléctrica

Si no se siguen (o se siguen de modo insuficiente) las indicaciones del fabricante en cuanto a seguridad eléctrica, pueden producirse choques eléctricos, incendios y/o lesiones graves con posible consecuencia de muerte.

1. Antes de la puesta en marcha del producto se deberá comprobar siempre que la tensión preseleccionada en el producto coincida con la de la red de alimentación eléctrica. Si es necesario modificar el ajuste de tensión, también se deberán cambiar en caso dado los fusibles correspondientes del producto.
2. Los productos de la clase de protección I con alimentación móvil y enchufe individual solamente podrán enchufarse a tomas de corriente con contacto de seguridad y con conductor de protección conectado.
3. Queda prohibida la interrupción intencionada del conductor de protección, tanto en la toma de corriente como en el mismo producto. La interrupción puede tener como consecuencia el riesgo de que el producto sea fuente de choques eléctricos. Si se utilizan cables alargadores o regletas de enchufe, deberá garantizarse la realización de un examen regular de los mismos en cuanto a su estado técnico de seguridad.
4. Si el producto no está equipado con un interruptor para desconectarlo de la red, o bien si el interruptor existente no resulta apropiado para la desconexión de la red, el enchufe del cable de conexión se deberá considerar como un dispositivo de desconexión.
   El dispositivo de desconexión se debe poder alcanzar fácilmente y debe estar siempre bien accesible. Si, p. ej., el enchufe de conexión a la red es el dispositivo de desconexión, la longitud del cable de conexión no debe superar 3 m).
   Los interruptores selectores o electrónicos no son aptos para el corte de la red eléctrica. Si se integran productos sin interruptor en bastidores o instalaciones, se deberá colocar el interruptor en el nivel de la instalación.
5. No utilice nunca el producto si está dañado el cable de conexión a red. Compruebe regularmente el correcto estado de los cables de conexión a red. Asegúrese, mediante las medidas de protección y de instalación adecuadas, de que el cable de conexión a red no pueda ser dañado o de que nadie pueda ser dañado por él, p. ej. al tropezar o por un choque eléctrico.

6. Solamente está permitido el funcionamiento en redes de alimentación TN/TT aseguradas con fusibles de 16 A como máximo (utilización de fusibles de mayor amperaje solo previa consulta con el grupo de empresas Rohde & Schwarz).
7. Nunca conecte el enchufe en tomas de corriente sucias o llenas de polvo. Introduzca el enchufe por completo y fuertemente en la toma de corriente. La no observación de estas medidas puede provocar chispas, fuego y/o lesiones.
8. No sobrecargue las tomas de corriente, los cables alargadores o las regletas de enchufe ya que esto podría causar fuego o choques eléctricos.
9. En las mediciones en circuitos de corriente con una tensión U_{eff} > 30 V se deberán tomar las medidas apropiadas para impedir cualquier peligro (p. ej. medios de medición adecuados, seguros, limitación de tensión, corte protector, aislamiento etc.).
10. Para la conexión con dispositivos informáticos como un PC o un ordenador industrial, debe comprobarse que éstos cumplan los estándares IEC60950-1/EN60950-1 o IEC61010-1/EN 61010-1 válidos en cada caso.
11. A menos que esté permitido expresamente, no retire nunca la tapa ni componentes de la carcasa mientras el producto esté en servicio. Esto pone a descubierto los cables y componentes eléctricos y puede causar lesiones, fuego o daños en el producto.
12. Si un producto se instala en un lugar fijo, se deberá primero conectar el conductor de protección fijo con el conductor de protección del producto antes de hacer cualquier otra conexión. La instalación y la conexión deberán ser efectuadas por un electricista especializado.
13. En el caso de dispositivos fijos que no estén provistos de fusibles, interruptor automático ni otros mecanismos de seguridad similares, el circuito de alimentación debe estar protegido de modo que todas las personas que puedan acceder al producto, así como el producto mismo, estén a salvo de posibles daños.
14. Todo producto debe estar protegido contra sobretensión (debida p. ej. a una caída del rayo) mediante los correspondientes sistemas de protección. Si no, el personal que lo utilice quedará expuesto al peligro de choque eléctrico.
15. No debe introducirse en los orificios de la caja del aparato ningún objeto que no esté destinado a ello. Esto puede producir cortocircuitos en el producto y/o puede causar choques eléctricos, fuego o lesiones.
16. Salvo indicación contraria, los productos no están impermeabilizados (ver también el capítulo "Estados operativos y posiciones de funcionamiento", punto 1). Por eso es necesario tomar las medidas necesarias para evitar la entrada de líquidos. En caso contrario, existe peligro de choque eléctrico para el usuario o de daños en el producto, que también pueden redundar en peligro para las personas.
17. No utilice el producto en condiciones en las que pueda producirse o ya se hayan producido condensaciones sobre el producto o en el interior de éste, como p. ej. al desplazarlo de un lugar frío a otro caliente. La entrada de agua aumenta el riesgo de choque eléctrico.
18. Antes de la limpieza, desconecte por completo el producto de la alimentación de tensión (p. ej. red de alimentación o batería). Realice la limpieza de los aparatos con un paño suave, que no se deshilache. No utilice bajo ningún concepto productos de limpieza químicos como alcohol, acetona o diluyentes para lacas nitrocelulósicas.

Funcionamiento

1. El uso del producto requiere instrucciones especiales y una alta concentración durante el manejo. Debe asegurarse que las personas que manejen el producto estén a la altura de los requerimientos necesarios en cuanto a aptitudes físicas, psíquicas y emocionales, ya que de otra manera no se pueden excluir lesiones o daños de objetos. El empresario u operador es responsable de seleccionar el personal usuario apto para el manejo del producto.
2. Antes de desplazar o transportar el producto, lea y tenga en cuenta el capítulo "Transporte".
3. Como con todo producto de fabricación industrial no puede quedar excluida en general la posibilidad de que se produzcan alergias provocadas por algunos materiales empleados —los llamados alérgenos (p. ej. el níquel)—. Si durante el manejo de productos Rohde & Schwarz se producen reacciones alérgicas, como p. ej. irritaciones cutáneas, estornudos continuos, enrojecimiento de la conjuntiva o dificultades respiratorias, debe avisarse inmediatamente a un médico para investigar las causas y evitar cualquier molestia o daño a la salud.
4. Antes de la manipulación mecánica y/o térmica o el desmontaje del producto, debe tenerse en cuenta imprescindiblemente el capítulo "Eliminación/protección del medio ambiente", punto 1.
5. Ciertos productos, como p. ej. las instalaciones de radiocomunicación RF, pueden a causa de su función natural, emitir una radiación electromagnética aumentada. Deben tomarse todas las medidas necesarias para la protección de las mujeres embarazadas. También las personas con marcapasos pueden correr peligro a causa de la radiación electromagnética. El empresario/operador tiene la obligación de evaluar y señalizar las áreas de trabajo en las que exista un riesgo elevado de exposición a radiaciones.
6. Tenga en cuenta que en caso de incendio pueden desprenderse del producto sustancias tóxicas (gases, líquidos etc.) que pueden generar daños a la salud. Por eso, en caso de incendio deben usarse medidas adecuadas, como p. ej. máscaras antigás e indumentaria de protección.
7. Los productos con láser están provistos de indicaciones de advertencia normalizadas en función de la clase de láser del que se trate. Los rayos láser pueden provocar daños de tipo biológico a causa de las propiedades de su radiación y debido a su concentración extrema de potencia electromagnética. En caso de que un producto Rohde & Schwarz contenga un producto láser (p. ej. un lector de CD/DVD), no debe usarse ninguna otra configuración o función aparte de las descritas en la documentación del producto, a fin de evitar lesiones (p. ej. debidas a irradiación láser).
8. Clases de compatibilidad electromagnética (conforme a EN 55011 / CISPR 11; y en analogía con EN 55022 / CISPR 22, EN 55032 / CISPR 32)
   - Aparato de clase A:
     Aparato adecuado para su uso en todos los entornos excepto en los residenciales y en aquellos conectados directamente a una red de distribución de baja tensión que suministra corriente a edificios residenciales.
     Nota: Los aparatos de clase A están destinados al uso en entornos industriales. Estos aparatos pueden causar perturbaciones radioeléctricas en entornos residenciales debido a posibles perturbaciones guiadas o radiadas. En este caso, se le podrá solicitar al operador que tome las medidas adecuadas para eliminar estas perturbaciones.
   - Aparato de clase B:
     Aparato adecuado para su uso en entornos residenciales, así como en aquellos conectados directamente a una red de distribución de baja tensión que suministra corriente a edificios residenciales.

Reparación y mantenimiento

1. El producto solamente debe ser abierto por personal especializado con autorización para ello. Antes de manipular el producto o abrirlo, es obligatorio desconectarlo de la tensión de alimentación, para evitar toda posibilidad de choque eléctrico.
2. El ajuste, el cambio de partes, el mantenimiento y la reparación deberán ser efectuadas solamente por electricistas autorizados por Rohde & Schwarz. Si se reponen partes con importancia para los aspectos de seguridad (p. ej. el enchufe, los transformadores o los fusibles), solamente podrán ser sustituidos por partes originales. Después de cada cambio de partes relevantes para la seguridad deberá realizarse un control de seguridad (control a primera vista, control del conductor de protección, medición de resistencia de aislamiento, medición de la corriente de fuga, control de funcionamiento). Con esto queda garantizada la seguridad del producto.

Baterías y acumuladores o celdas

Si no se siguen (o se siguen de modo insuficiente) las indicaciones en cuanto a las baterías y acumuladores o celdas, pueden producirse explosiones, incendios y/o lesiones graves con posible consecuencia de muerte. El manejo de baterías y acumuladores con electrolitos alcalinos (p. ej. celdas de litio) debe seguir el estándar EN 62133.

1. No deben desmontarse, abrirse ni triturarse las celdas.
2. Las celdas o baterías no deben someterse a calor ni fuego. Debe evitarse el almacenamiento a la luz directa del sol. Las celdas y baterías deben mantenerse limpias y secas. Limpiar las conexiones sucias con un paño seco y limpio.
3. Las celdas o baterías no deben cortocircuitarse. Es peligroso almacenar las celdas o baterías en estuches o cajones en cuyo interior puedan cortocircuitarse por contacto recíproco o por contacto con otros materiales conductores. No deben extraerse las celdas o baterías de sus embalajes originales hasta el momento en que vayan a utilizarse.
4. Las celdas o baterías no deben someterse a impactos mecánicos fuertes indebidos.
5. En caso de falta de estanqueidad de una celda, el líquido vertido no debe entrar en contacto con la piel ni los ojos. Si se produce contacto, lavar con agua abundante la zona afectada y avisar a un médico.
6. En caso de cambio o recarga inadecuados, las celdas o baterías que contienen electrolitos alcalinos (p. ej. las celdas de litio) pueden explotar. Para garantizar la seguridad del producto, las celdas o baterías solo deben ser sustituidas por el tipo Rohde & Schwarz correspondiente (ver lista de recambios).
7. Las baterías y celdas deben reciclarse y no deben tirarse a la basura doméstica. Las baterías o acumuladores que contienen plomo, mercurio o cadmio deben tratarse como residuos especiales. Respete en esta relación las normas nacionales de eliminación y reciclaje.

Transporte

1. El producto puede tener un peso elevado. Por eso es necesario desplazarlo o transportarlo con precaución y, si es necesario, usando un sistema de elevación adecuado (p. ej. una carretilla elevadora), a fin de evitar lesiones en la espalda u otros daños personales.

2. Las asas instaladas en los productos sirven solamente de ayuda para el transporte del producto por personas. Por eso no está permitido utilizar las asas para la sujeción en o sobre medios de transporte como p. ej. grúas, carretillas elevadoras de horquilla, carros etc. Es responsabilidad suya fijar los productos de manera segura a los medios de transporte o elevación. Para evitar daños personales o daños en el producto, siga las instrucciones de seguridad del fabricante del medio de transporte o elevación utilizado.
3. Si se utiliza el producto dentro de un vehículo, recae de manera exclusiva en el conductor la responsabilidad de conducir el vehículo de manera segura y adecuada. El fabricante no asumirá ninguna responsabilidad por accidentes o colisiones. No utilice nunca el producto dentro de un vehículo en movimiento si esto pudiera distraer al conductor. Asegure el producto dentro del vehículo debidamente para evitar, en caso de un accidente, lesiones u otra clase de daños.

Eliminación/protección del medio ambiente

1. Los dispositivos marcados contienen una batería o un acumulador que no se debe desechar con los residuos domésticos sin clasificar, sino que debe ser recogido por separado. La eliminación se debe efectuar exclusivamente a través de un punto de recogida apropiado o del servicio de atención al cliente de Rohde & Schwarz.
2. Los dispositivos eléctricos usados no se deben desechar con los residuos domésticos sin clasificar, sino que deben ser recogidos por separado.
   Rohde & Schwarz GmbH & Co.KG ha elaborado un concepto de eliminación de residuos y asume plenamente los deberes de recogida y eliminación para los fabricantes dentro de la UE. Para desechar el producto de manera respetuosa con el medio ambiente, diríjase a su servicio de atención al cliente de Rohde & Schwarz.
3. Si se trabaja de manera mecánica y/o térmica cualquier producto o componente más allá del funcionamiento previsto, pueden liberarse sustancias peligrosas (polvos con contenido de metales pesados como p. ej. plomo, berilio o níquel). Por eso el producto solo debe ser desmontado por personal especializado con formación adecuada. Un desmontaje inadecuado puede ocasionar daños para la salud. Se deben tener en cuenta las directivas nacionales referentes a la eliminación de residuos.
4. En caso de que durante el trato del producto se formen sustancias peligrosas o combustibles que deban tratarse como residuos especiales (p. ej. refrigerantes o aceites de motor con intervalos de cambio definidos), deben tenerse en cuenta las indicaciones de seguridad del fabricante de dichas sustancias y las normas regionales de eliminación de residuos. Tenga en cuenta también en caso necesario las indicaciones de seguridad especiales contenidas en la documentación del producto. La eliminación incorrecta de sustancias peligrosas o combustibles puede causar daños a la salud o daños al medio ambiente.

Se puede encontrar más información sobre la protección del medio ambiente en la página web de Rohde & Schwarz.

Quality management and environmental management

Certified Quality System
ISO 9001

Certified Environmental System
ISO 14001

Sehr geehrter Kunde,
Sie haben sich für den Kauf eines Rohde&Schwarz Produktes entschieden. Sie erhalten damit ein nach modernsten Fertigungsmethoden hergestelltes Produkt. Es wurde nach den Regeln unserer Qualitäts- und Umweltmanagementsysteme entwickelt, gefertigt und geprüft. Rohde&Schwarz ist unter anderem nach den Managementsystemen ISO 9001 und ISO 14001 zertifiziert.

Der Umwelt verpflichtet
- Energie-effiziente, RoHS-konforme Produkte
- Kontinuierliche Weiterentwicklung nachhaltiger Umweltkonzepte
- ISO 14001-zertifiziertes Umweltmanagementsystem

Dear customer,
You have decided to buy a Rohde&Schwarz product. This product has been manufactured using the most advanced methods. It was developed, manufactured and tested in compliance with our quality management and environmental management systems. Rohde&Schwarz has been certified, for example, according to the ISO 9001 and ISO 14001 management systems.

Environmental commitment
- Energy-efficient products
- Continuous improvement in environmental sustainability
- ISO 14001-certified environmental management system

Cher client,
Vous avez choisi d'acheter un produit Rohde&Schwarz. Vous disposez donc d'un produit fabriqué d'après les méthodes les plus avancées. Le développement, la fabrication et les tests de ce produit ont été effectués selon nos systèmes de management de qualité et de management environnemental. La société Rohde&Schwarz a été homologuée, entre autres, conformément aux systèmes de management ISO 9001 et ISO 14001.

Engagement écologique
- Produits à efficience énergétique
- Amélioration continue de la durabilité environnementale
- Système de management environnemental certifié selon ISO 14001

ROHDE&SCHWARZ
1171.0200.11 V 05.01
1171020011

Customer Support

Technical support – where and when you need it

For quick, expert help with any Rohde & Schwarz equipment, contact one of our Customer Support Centers. A team of highly qualified engineers provides telephone support and will work with you to find a solution to your query on any aspect of the operation, programming or applications of Rohde & Schwarz equipment.

Up-to-date information and upgrades

To keep your instrument up-to-date and to be informed about new application notes related to your instrument, please send an e-mail to the Customer Support Center stating your instrument and your wish. We will take care that you will get the right information.

Europe, Africa, Middle East

Phone +49 89 4129 12345
customersupport@rohde-schwarz.com

North America

Phone 1-888-TEST-RSA (1-888-837-8772)
customer.support@rsa.rohde-schwarz.com

Latin America

Phone +1-410-910-7988
customersupport.la@rohde-schwarz.com

Asia/Pacific

Phone +65 65 13 04 88
customersupport.asia@rohde-schwarz.com

China

Phone +86-800-810-8228 /
+86-400-650-5896
customersupport.china@rohde-schwarz.com

ROHDE&SCHWARZ

1171.0200.22-06.00

Contents

1 Applications

1.1 General

This manual describes the function and operation of the ROHDE&SCHWARZ digital functional test module R&S TS-PIO4 for use in the test system versatile platform R&S CompactTSVP. The hardware is implemented as a CompactPCI board that occupies only one slot on the front side of the TSVP.

The R&S TS-PIO4 digital functional test module is used wherever digital circuits are tested by flexibly programmable static or dynamic stimulation and the response is recorded.

Deterministic, simultaneous stimulation and recording of digital signals makes it possible to simulate operating conditions in a manner that is very close to reality. A large local memory pool and an independent sequence controller are available on the module to ensure that the precise and predictable time response can be maintained for output, recording and analysis of the bit patterns. Extensive trigger options via the PXI trigger bus enable synchronization with additional R&S TS-PIO4 modules or other measurement and stimulus modules. This makes it possible to increase the number of digital channels in an application. Measurements in which signals are to be recorded synchronously are also possible.

Output levels and input thresholds can be programmed in 8 ports each with 4 channels. This allows for optimum adaptability to the requirements of different logic families. The effect of interference signals in the test setup can be minimized by adjusting the hysteresis for the input thresholds.

Safety circuits to protect against short circuits, countervoltages and overvoltages contribute to the robustness of the R&S TS-PIO4 digital functional test module. Due to the extremely space-saving design of the I/O safety circuit and signal conditioning unit, the R&S TS-PIO4 occupies just one CompactPCI/PXI slot. This makes it possible to set up very high-performance and compact measuring systems.

The R&S TS-PIO4 digital functional test module is intended for the R&S CompactTSVP test platform. The module is controlled by the CompactPCI bus.

A soft panel is available for operation of the module. An IVI-C driver is provided to allow the module to be used under software control.

1.2 Features

Features of the R&S TS-PIO4 digital functional test module.

- 32 digital inputs and 32 digital outputs divided into 8 ports each with 4 channels
- Programmable output level range from –6 V to 10 V, output impedance 30 Ω (typ.)
- Resolution 14 bits
- Output current up to 150 mA per channel, 350 mA per port

- Tri-state control in dynamic and static mode
- Two programmable input thresholds per port for hysteresis or level monitoring
- Inputs can be connected to outputs
- Pattern rate up to 40 MS/s per channel x 12.5 nsec. resolution
- Memory depth 2 Msample (32-bit data)
- External clock input
- Synchronization/triggering via PXI trigger bus and via XTI (TTL)
- Timer/counter function
- FPGA-based flexibility
- Standalone self-test capability
- LabWindows IVI-C driver available
- Used in R&S CompactTSVP

1.3 Possible Applications

The R&S TS-PIO4 digital functional test module is used for testing digital modules or devices. This type of functional test is used to check the overall operation of a digital circuit under conditions that are as close to reality as possible. The module does this by applying digital input patterns, measuring the output signals and comparing them with the nominal values.

The R&S TS-PIO4 digital functional test module can be used for tasks such as the following:

- Digital functional test (low-speed/high-speed, IO control)
- Bit pattern stimulation (low-speed/high-speed, digital buses)
- Bit pattern measurement (low-speed/high-speed)
- Monitoring of changes in level state (pattern trigger)
- Digital functional test at component level (no node-forcing or backdriving capability)
- Downloads, e.g. for flash components, serial and parallel

A typical functional test consists of the following elements:

- Adaptation of the pin electronics to the UUT environment (logic level and logic family)
- Definition of the sensor strobe (timing)
- Definition of the stimulus and measuring behavior
- Evaluation of the test result

1.3.1 Digital Functional Test – Static

Characteristics for which correct interaction of the logic modules is more important than the verification of time-critical limits are tested in the static digital functional test. The

application outputs the patterns to be stimulated, reads in the UUT response via the module inputs and compares it with the expected response. Comparison with the nominal values then results in a PASS or FAIL statement. The comparison takes place in the application. As the time response (length of time that a pattern is available; sampling time point of the inputs) is controlled primarily by the host computer, this test cannot be used to test time-critical or chronologically precisely defined sequences. With the static test, the order of the sequence is deterministic, i.e. the chronological order of the patterns as well as the read-in procedure is predictable. How long the interval from one pattern to the next pattern will be and how large the interval from application of a pattern to reading in of the data will be are, however, not predictable. Here, the operating system of the host computer (task switching, etc.) influences the runtimes in a non-predictable way.

1.3.2 Digital Functional Test – Dynamic

In the real-time test, overall operation of the digital section of a UUT is tested under operating conditions that are as close to reality as possible. For this purpose, digital patterns (vectors and pattern sets) with a precisely defined, usually high clock rate and precise time response are applied at the UUT connections and the responses recorded. Recording is also performed with a precisely defined, reproducible time response. The basic requirement for exact, predictable timing is that the patterns for the stimulus are stored in the memories "downstream of the drivers" and can be processed with a defined time response and at a high speed. To enable this, the module has its own sequence controller in the FPGA. The recorded input data is also initially stored in a memory on the module and later retrieved by the application on the host computer and evaluated for a PASS/FAIL decision.

2 View

Figure 2-1 shows the R&S TS-PIO4 digital functional test module (without heat sink).

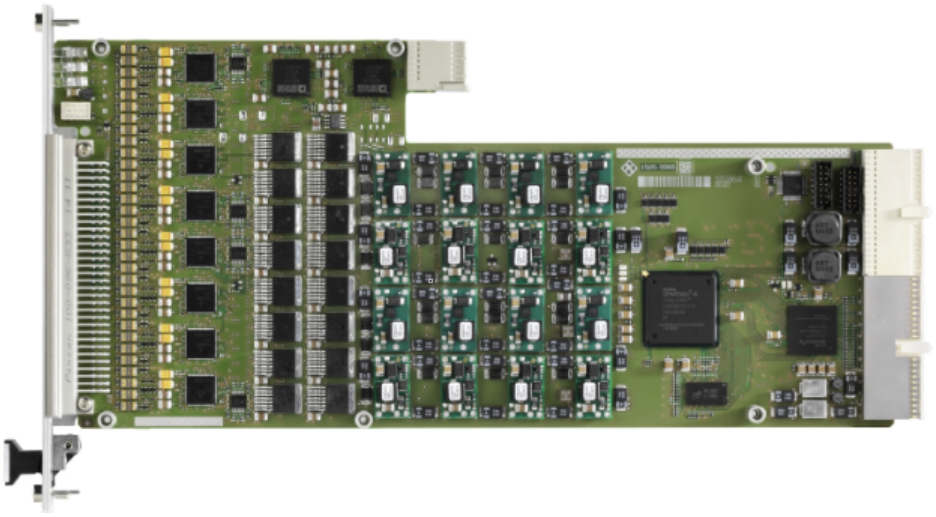


Figure 2-1: View of module

3 Block Diagrams

This section contains a functional block diagram of the R&S TS-PIO4 module as well as a detailed block diagram.

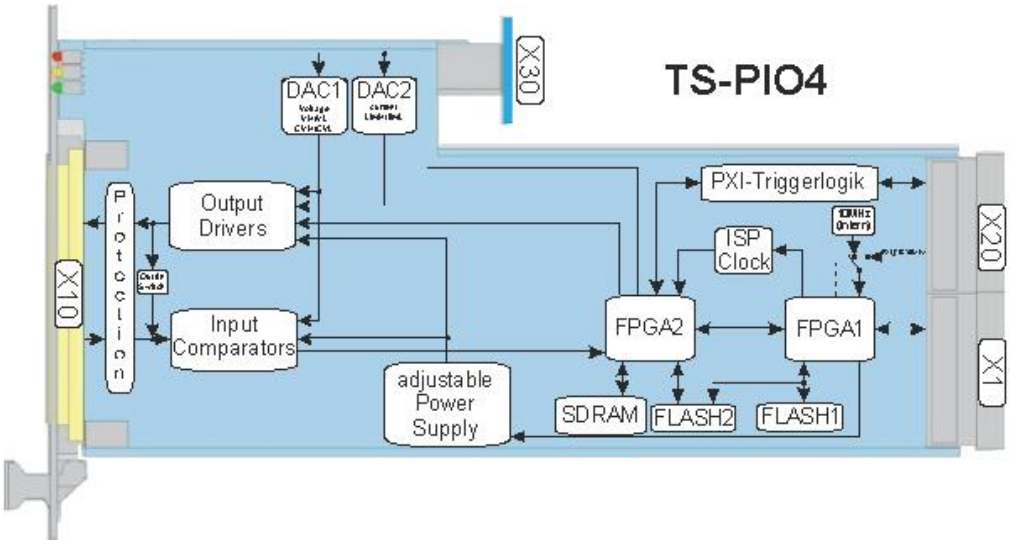


Figure 3-1: Functional block diagram of R&S TS-PIO4 module

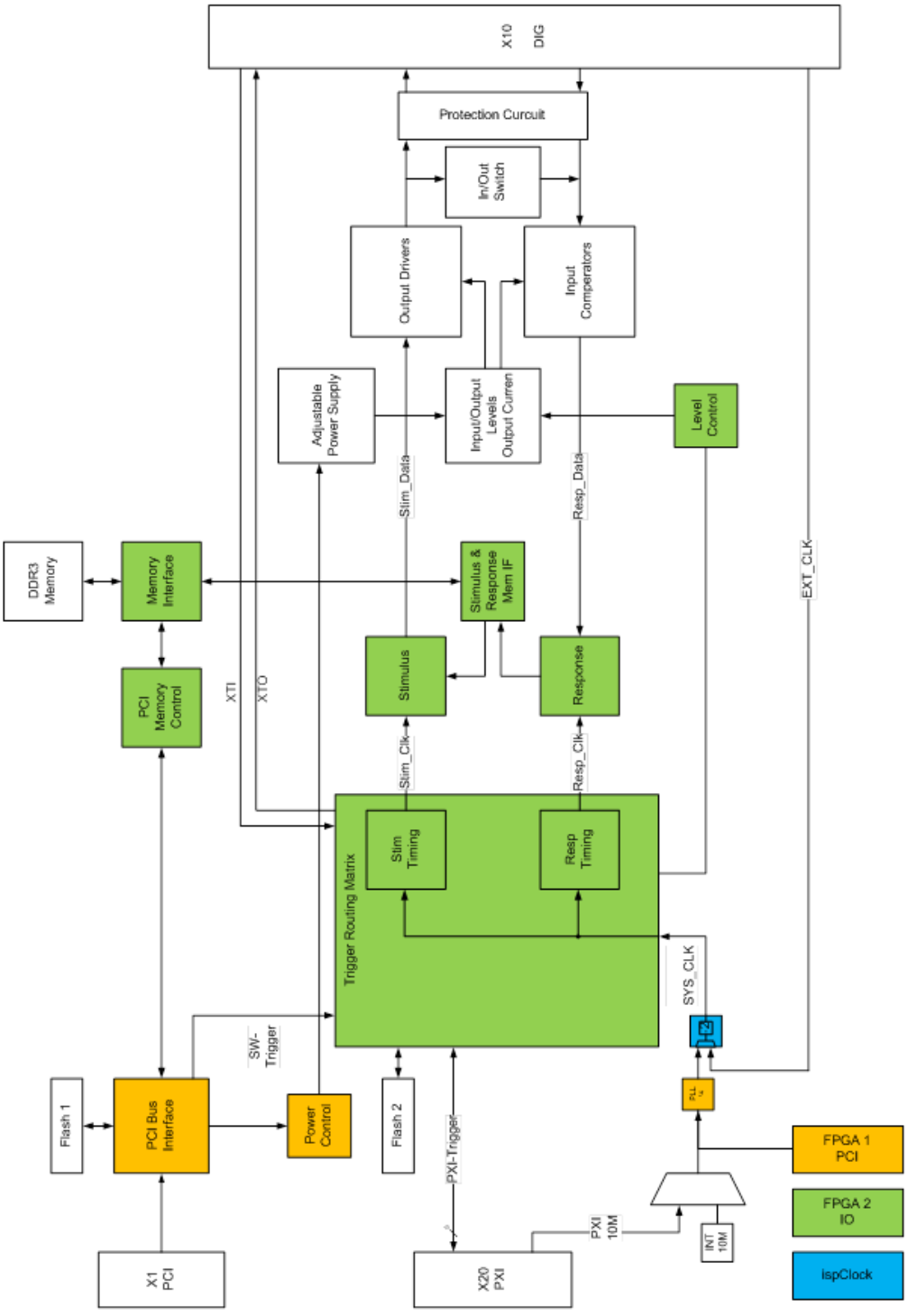


Figure 3-2: Detailed block diagram of R&S TS-PIO4 module

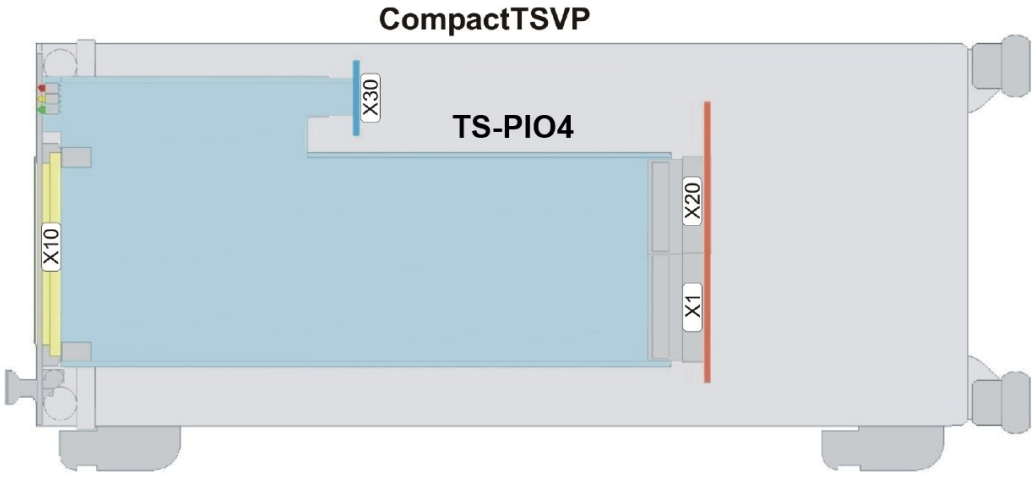


Figure 3-3: R&S TS-PIO4 in R&S CompactTSVP

4 Design

4.1 Mechanical Design

The R&S TS-PIO4 module is designed as a long cPCI plug-in module for mounting in the front of the R&S CompactTSVP.

The height of the module's board is 3 HU (134 mm). The front panel has a locating pin to ensure that the module is correctly inserted into the R&S CompactTSVP. The module is secured using the two fastening screws on the front panel.

The front interface X10 is used for connecting UUTs.

Connector X30 is solely intended to provide mechanical stability. The contacts of the connector are not used.

Interfaces X20/X1 connect the R&S TS-PIO4 module to the cPCI backplane/PXI control backplane.

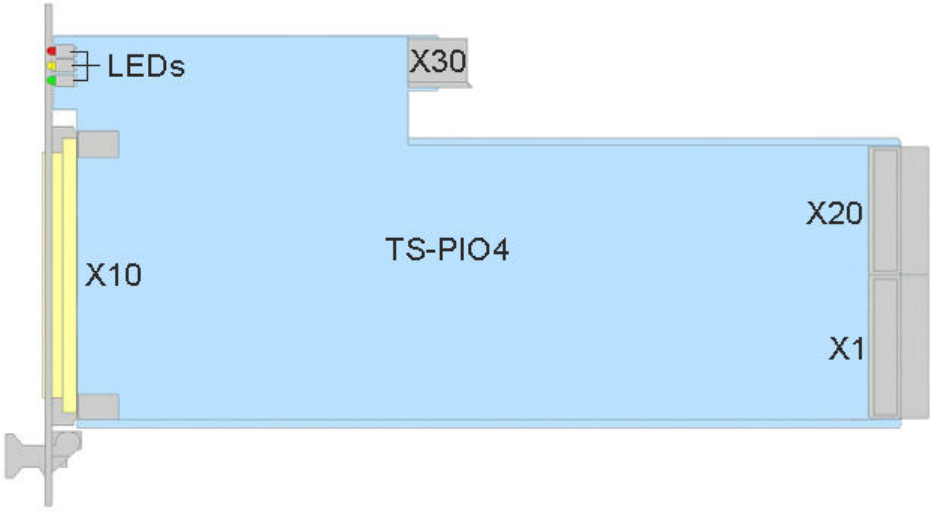


Figure 4-1: Arrangement of interfaces on R&S TS-PIO4 module

4.2 Interfaces

Name	Use
X1	cPCI bus backplane
X10	Unit under test (UUT)
X20	Backplane extension (PXI), rear I/O
X30	Analog bus (not used)

A detailed interface description with signal assignment to the connectors can be found in Chapter 9, "Interface Description", on page 79.

4.3 Display Elements

On the front panel of the R&S TS-PIO4 are three light emitting diodes (LEDs) which have the following meaning:

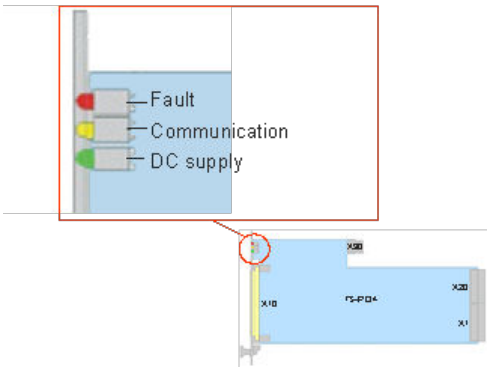


Figure 4-2: LED indicators of R&S TS-PIO4

LED	Description
ERR (red)	Error condition: Lights up if, after the supply voltage is switched on, a fault is detected on the R&S TS-PIO4 module during the power-on test.
COM (yellow)	Communication: Lights up when data is being exchanged via the interface.
PWR (green)	Supply voltage: Lights up when all the necessary supply voltages are present.

5 Functional Description

5.1 Overview

5.1.1 General

The R&S TS-PIO4 digital functional test module provides 8 ports each with 4 digital input pins and output pins. For this purpose, the module has unmultiplexed digital pins, i.e. behind each pin is a separate digital channel. Furthermore, the output voltages of the drivers as well as the comparator thresholds of the inputs can, within certain limits, be freely programmed on the module. This allows the appropriate logic level to be generated for practically every application. All settings as well as clock generation take place on the module itself, which means that no additional stimulus modules are necessary.

The usable level range depends on the configuration of the driver references. The overall level range is divided into 4 sections. The driver levels are generated from the CPCI supply (+5 V and +3.3 V).

The application software must set an appropriate voltage range before the output voltages and comparator thresholds can be configured. Here, the device driver performs a plausibility check and prevents invalid or damaging values from being set.

Table 5-1: Voltage ranges

Range	VL[1]	VH[2]	CVL[3] / CVH[4]
3	+1.5 V to +6.0 V	+1.5 V to +10.0 V	+1.5 V to +7.1 V
2	−1.8 V to +2.7 V	−1.8 V to +10.0 V	−1.8 V to +7.1 V
1	−3.5 V to +1.0 V	−3.5 V to +10.0 V	−3.5 V to +7.1 V
0	−6.0 V to −1.5 V	−6.0 V to +8.0 V	−6.0 V to +5.1 V

[1]VL: Output level Low

[2]VH: Output level High

[3]CVL: Comparator level Low

[4]CVH: Comparator level High

If required, all outputs (OUTx) can be connected to their corresponding input (INx) and therefore stimulate, measure and monitor the driving of signals. All driver channels can be set to the high-impedance state (TRI-STATE).

Timing control of data output and reading in of response signals takes place on the module controlled by FPGAs.

5.1.2 Ports

Due to the structures in the hardware, the digital inputs and outputs are divided into so-called ports.

The R&S TS-PIO4 module provides 32 outputs (OUT1 to OUT32) and 32 inputs (IN1 to IN32). Each port (PORT0 to PORT7) is made up of 4 channels.

Many functions of the driver refer to this port structure.

Table 5-2: Port structure

Port	Channel OUTx / INx	Bit mask
0	1…4	RSPIO4_MASK_PORT0
1	5…8	RSPIO4_MASK_PORT1
2	9…12	RSPIO4_MASK_PORT2
3	13…16	RSPIO4_MASK_PORT3
4	17…20	RSPIO4_MASK_PORT4
5	21…24	RSPIO4_MASK_PORT5
6	25…28	RSPIO4_MASK_PORT6
7	29…32	RSPIO4_MASK_PORT7

5.1.3 Memories

5.1.3.1 Memory Management in Driver

If required, the digital channels can be configured in different combinations for simultaneous static and dynamic operation. Various formats (data types) are therefore available for the dynamic data records.

Stimulus data records which are to be loaded to the module using `rspio4_LoadData` are first stored in the memory management. For each data record, the driver assigns an ID using which this data record can then, if required, be identified and executed.

If access takes place using the driver functions compliant with IVI Digital, this runs in the background and can be ignored by the user. The appropriate data formats are then also selected automatically.

The data is interpreted in the driver according to the mode to which the module was set beforehand by means of the `rspio4_ConfigureStimMode()` and `rspio4_ConfigureRespMode()` functions.

For reasons of backward compatibility, the attribute `RSPIO4_ATTR_PORT_HANDLING` can be set to the value *RSPIO4_PORT_HANDLING_PDFT* using the `rspio4_ConfigurePortHandling` function. In this case, the driver software emulates a module with 4 ports each with 8 channels. With this setting, the tri-state information in the data structures is interpreted on a port-specific basis.

The file `rspio4.h` provides these data types. For details, see `rspio4.h` as well as the help file of the driver.

Table 5-3: Data types

Mode (see `rspio4.h`)	**Data type** (see `rspio4.h`)
RSPIO4_VAL_CTRL_STATIC	
RSPIO4_VAL_CTRL_4BIT	RSPIO4_DATA_4BIT (stim. and resp.)
RSPIO4_VAL_CTRL_8BIT	RSPIO4_DATA_8BIT (stim. and resp.)
RSPIO4_VAL_CTRL_16BIT	RSPIO4_DATA_16BIT (stim. and resp.)
RSPIO4_VAL_CTRL_32BIT	RSPIO4_DATA_32BIT (stim. and resp.)
RSPIO4_VAL_CTRL_4BIT_TRISTATE	RSPIO4_DATA_4BIT_TRISTATE (stim.) RSPIO4_DATA_4BIT (resp.)
RSPIO4_VAL_CTRL_8BIT_TRISTATE	RSPIO4_DATA_8BIT_TRISTATE (stim.) RSPIO4_DATA_8BIT (resp.)
RSPIO4_VAL_CTRL_16BIT_TRISTATE	RSPIO4_DATA_16BIT_TRISTATE (stim.) RSPIO4_VAL_CTRL_16BIT (resp.)
RSPIO4_VAL_CTRL_32BIT_TRISTATE	RSPIO4_DATA_32BIT_TRISTATE (stim.) RSPIO4_VAL_CTRL_32BIT (resp.)

5.1.3.2 Stimulus Memory

2 Msample memories for stimulus data are available in the module. This means that max. one pattern set with 2 x 1024 x 1024 data vectors plus 2 x 1024 x 1024 enable vectors can be defined and loaded to the module.

The data records are first transferred to the stimulus memory using the `spio4_LoadData()` function. This function returns an ID. This ID can then be used to execute the data record in the stimulus memory.

Depending on the function used to execute a pattern set, this takes place automatically or must be triggered manually (e.g. IVI Digital functions perform this sequence completely in the background).

The `rspio4_DiscardData()` function is used to remove the data record from the stimulus memory.

5.1.3.3 Response Memory

The recorded response data is also stored in a 2 Msample memory. This memory always contains the results of the high comparators and the low comparators alternately in a 32-bit value.

The default functions evaluate these results according to the selected comparator mode:

- COMP window comparator
- HYST hysteresis

To allow information with regard to an input voltage in a prohibited range to be received, functions which evaluate and return the validity of the data of each channel have been added to the driver. (See also Chapter 5.3.3, "Configuration of Input Channels", on page 30.)

5.1.4 Static Digital Test

With the static digital test, the sequence for applying patterns at the outputs as well as for reading in the inputs is checked from the control computer. As a result, the duration of the individual patterns as well as the sampling time point for the measurement relative to the beginning of the pattern can never be precisely predicted because the host computer and its operating system are influencing factors here.

The static digital test is therefore suitable for checking the logical interaction of components in order to check voltage thresholds and other sequences in cases where the verification and the precise observance of time conditions are not relevant.

With R&S TS-PIO4, the static digital test can be performed using IVI Digital functions or the low-level driver functions.

5.1.5 Dynamic Digital Test

In the simplest case, the pattern period is identical for output (stimulus) and recording (response). Stimulus and response are started by the same trigger and run synchronously to each other.

Normally, the UUT response must be measured with a delay relative to the beginning of the pattern. The delay (Response Delay) can be set between 0 s and the pattern period. This delay is defined by the application such that stable data from the UUT is present at the time of sampling. This value is ultimately determined by the delay times in the UUT. If the delay is greater than the pattern period, then sampling already takes place within the next period of the stimulus patterns. This too may be advisable in certain applications.

The test is started by software or hardware triggers. The patterns are output at a fixed clock rate. The pattern duration is the time during which a pattern of a pattern set is available at the outputs. The responses from the UUT are usually also recorded at the inputs during this time.

The "Response Delay" is the time offset between the beginning of a pattern and sampling of the data at the inputs. The pattern rate is the reciprocal of the pattern period.

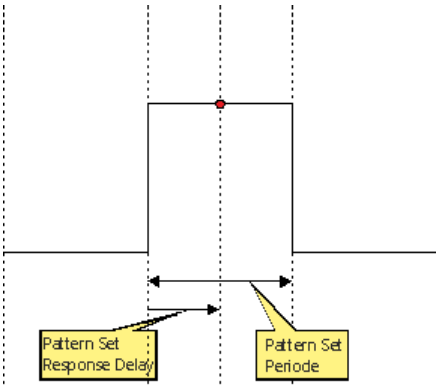


Figure 5-1: "Pattern Set Period" and "Response Delay"

A pattern set is a quantity of patterns (vectors) that are processed in a sequence after a trigger event has been received. Processing is started by a software function or a hardware trigger. Execution takes place in real time and is not affected by the operating system of the control computer. The software is able to query whether execution is still in progress, to interrupt the current execution or to wait until execution has finished.

5.2 Programming of Digital Tests

TSVP is an open platform with regard to hardware and software. Typically, the software consists of a number of drivers and libraries that are called from a test program or sequencer created by the user.

Generally there are two ways to access TSVP modules.

- Instrument driver
- High-level GTSL libraries

Device driver functions provide access to all hardware options. In contrast, symbolic names and cross-module functionality are not supported.

The functions in the device driver provide two interfaces for programming digital tests:

- Digital test with low-level functions
- Digital test with IVI Digital functions

High-level libraries provide a standardized programming interface and allow certain abstractions with respect to the underlying hardware, e.g. symbolic signal names and the handling of multiple parallel modules.

The GTSL library "DIO Manager" (`DIOMGR.DLL`) is provided for operation of the digital functions of the module.

5.2.1 Digital Tests with Device Driver Functions

5.2.1.1 Initialization

- `rspio4_init`
- `rspio4_InitWithOptions`
- `rspio4_close`

5.2.1.2 Auxiliary Functions

- `rspio4_reset`
- `rspio4_LockSession`
- `rspio4_UnlockSession`
- `rspio4_self_test`
- `rspio4_revision_query`

5.2.1.3 Error Queries

- `rspio4_error_query`
- `rspio4_GetErrorInfo`
- `rspio4_ClearErrorInfo`
- `rspio4_error_message`

5.2.1.4 Functions of IVI Switch Component

On the R&S TS-PIO4, there are only the two channels "GND" and "GNDNO" that can be switched using the "IVI switch functions". The GND relay is used for this purpose.

- `rspio4_Connect`
- `rspio4_Disconnect`
- `rspio4_DisconnectAll`
- `rspio4_GetPath`
- `rspio4_SetPath`
- `rspio4_CanConnect`
- `rspio4_IsDebounced`
- `rspio4_WaitForDebounce`

5.2.1.5 Digital Test with Low-Level Driver Functions

For optimum utilization of the board's performance, it is recommended to use the digital test with low-level functions.

Static digital test with low-level driver functions

- `rspio4_SetDoutState`
- `rspio4_SetDoutPort`
- `rspio4_ConnectInOut`
- `rspio4_GetDinState`
- `rspio4_GetDinHighAndLowComp`

Dynamic digital test with low-level driver functions

Some of these functions are also required for configuring the default settings for dynamic tests compliant with IVI Digital (see Chapter 7.3.4, "Static Pattern Output with Low-Level Driver Functions", on page 65).

- `rspio4_ConfigDinComparator`
- `rspio4_ConfigureStimMode`
- `rspio4_ConfigureRespMode`
- `rspio4_ConfigureStimTiming`
- `rspio4_ConfigureRespTiming`
- `rspio4_LoadData`
- `rspio4_LoadStimBuffer`
- `rspio4_AppendStimBuffer`
- `rspio4_ExecutePattern`
- `rspio4_AbortExecution`
- `rspio4_FetchPatternResponseData`
- `rspio4_FetchPatternResponseDataFull`
- `rspio4_DiscardData`

5.2.1.6 Digital Test Compliant with IVI Digital

IVI Digital is a standard for the actuation of digital test instruments. The driver provides calls that are compliant with IVI Digital. IVI Digital supports both the static and the dynamic digital test.

Use of the IVI Digital functions requires an extremely high computing overhead on the host side and is relatively inflexible with regard to the hardware because IVI Digital is a standardization and cannot take performance-boosting features of the hardware into consideration. For example, splitting into static and dynamic channels is not possible.

Functions for digital test compliant with IVI Digital

- `rspio4_ClearPattern`
- `rspio4_ConfigureChannelOpcode`
- `rspio4_ConfigureMode`
- `rspio4_ConfigureGroupOpcode`
- `rspio4_ConfigureLargeGroupOpcode`

- rspio4_CreatePattern
- rspio4_GetChannelName
- rspio4_GetChannelOpcode
- rspio4_GetGroupOpcode
- rspio4_ConfigureStaticResponseDelay
- rspio4_ExecuteStaticPattern
- rspio4_FetchStaticChannelOpcode
- rspio4_GetStaticChannelName
- rspio4_FetchStaticChannelListResults
- rspio4_FetchStaticChannelResult
- rspio4_FetchStaticChannelData
- rspio4_FetchStaticChannelListData
- rspio4_AbortPatternSet
- rspio4_BeginPatternSetLoading
- rspio4_ClearPatternSet
- rspio4_ConfigurePatternSetMaxTime
- rspio4_ConfigurePatternSetTiming
- rspio4_CreatePatternSet
- rspio4_ExecutePatternSet
- rspio4_FetchPatternSetResult
- rspio4_GetDynamicChannelName
- rspio4_GetDynamicPatternCount
- rspio4_FetchDynamicChannelOpcode
- rspio4_GetPatternSetCount
- rspio4_GetPatternSetExecutedPatternCount
- rspio4_GetPatternSetLoadedPatternCount
- rspio4_GetPatternSetName
- rspio4_InitiatePatternSet
- rspio4_LoadDynamicPattern
- rspio4_WaitUntilPatternSetComplete
- rspio4_FetchDynamicChannelListResults
- rspio4_FetchDynamicChannelListPatternResults
- rspio4_FetchDynamicChannelResult
- rspio4_FetchDynamicPatternResult
- rspio4_FetchDynamicPatternListResults
- rspio4_FetchDynamicChannelData
- rspio4_FetchDynamicChannelListData
- rspio4_FetchDynamicChannelListPatternData

Static digital test compliant with IVI Digital

Here too static tests are possible, i.e. tests in which the host computer is responsible for control and which, as a result, do not have a fully defined time response. See the example in Chapter 7.3.3, "Static Pattern Execution with IVI Digital", on page 58.

Dynamic digital test compliant with IVI Digital

As shown in the example for the static digital test compliant with IVI Digital, a so-called op code is generated for each channel ("OUT1" to "OUT32", "IN1" to "IN32"). This op code controls whether a channel is to drive high or low or whether it is to switch to tri-state. The pattern generated in this way is then executed immediately and is available at the outputs.

In the case of the dynamic digital test compliant with IVI Digital, the generated patterns are saved in a pattern set.

After a timing has been defined, the complete pattern set is executed.

5.2.2 Digital Test with DIO Manager

Each pattern set is stored in its own file. This file can be edited manually using a test editor.

The file containing the pattern set is loaded during runtime to one or more R&S TS-PIO4 modules and then executed. The results can be read back using function calls. Alternatively, the results can also be stored in a file with the same format. Differences between the expected and measured behavior can then be easily identified by comparing the files.

See also the example in Chapter 7.3.1, "Digital Test with "DIO Manager" Library", on page 44.

5.2.2.1 File Format

The file format was originally defined by the Altera Quartus Waveform Generator.

```
GROUP CREATE bus = bus[7] bus[6] bus[5] bus[4] bus[3] bus[2] bus[1] bus[0];
INPUTS N14CR0805170 E_COM E_EC1 E_EC2 E_EC3 bus;
OUTPUTS N13HCPN31500 N13HCPM31500 N13HCPO31500 N13HCPL31500;
UNIT ns;
RADIX HEX;
PATTERN
    0.0> 0 0 0 0 0 00 = X X X X
 1000.0> 1 0 0 0 0 01 = X X X X
 2000.0> Z Z Z Z Z 02 = X X X X
 3000.0> 0 0 0 0 0 03 = X X X X
 4000.0> 1 0 0 0 0 04 = X X X X
 5000.0> Z Z Z Z Z 05 = X X X X
 6000.0> 0 0 0 0 0 06 = 1 X X X
 7000.0> 0 1 0 0 0 07 = 0 X X X
 8000.0> Z Z Z Z Z 08 = X X X X
```

```
 9000.0> 0 0 0 0 0 09 = X 1 X X
10000.0> 0 0 0 0 1 0A = X 0 X X
11000.0> Z Z Z Z Z 0B = X X X X
12000.0> 0 0 0 0 0 0C = X X 1 X
13000.0> 0 0 0 1 0 0D = X X 0 X
14000.0> Z Z Z Z Z 0E = X X X X
15000.0> 0 0 0 0 0 0F = X X X 1
16000.0> 0 0 1 0 0 10 = X X X 0
17000.0> Z Z Z Z Z 11 = X X X X
18000.0> X X X X X X = X X X X
;
```

```
GROUP CREATE bus = bus[7] bus[6] bus[5] bus[4] bus[3] bus[2]
bus[1] bus[0];
```

This instruction is optional. It is used to group pins into buses. It is also possible to define multiple buses.

```
INPUTS N14CR0805170 E_COM E_EC1 E_EC2 E_EC3 bus;
```

```
OUTPUTS N13HCPN31500 N13HCPM31500 N13HPCO31500 N13HCPL31500;
```

The INPUTS and OUTPUTS instructions define the channel names or group names. Here it is important that INPUTS are inputs of the UUT and therefore correspond to OUT1 to OUT32 of the R&S TS-PIO4 module. OUTPUTs are UUT outputs and therefore correspond to the inputs IN1 to IN32 of the module.

```
UNIT ns;
```

Defines the unit of the timestamp used in the file.

```
RADIX HEX;
```

Defines the base for the values in the case of buses. The DIO manager supports HEX only.

```
PATTERN
```

```
0.0> 0 Z Z Z Z 00 = X X X X;
```

.....

All lines following PATTERN define this pattern set. Each line represents a pattern at a certain time. The last line is ignored and is used only as end identification. All channels are X. The timestamps do not have to be equidistant.

Value	for INPUTS	for OUTPUTS
0	drive low	expect low
1	drive high	expect high
Z	high impedance	(not used)
X	(not used)	don't care

A hexadecimal value is entered for groups. The special cases X and Z indicate that all channels of the groups concerned assume this state.

5.2.2.2 Configuration of DIO Manager

The waveform file contains the logical name of the inputs and outputs. Assignment of physical channels on the R&S TS-PIO4 modules to logical names takes place in the file `application.ini`.

```
[bench->823916]
DIODevice = device->pio4
DIOChannelTable = io_channel->823916

[io_channel->823916]

E_COM          = pio4!out1
E_EC1          = pio4!out2
E_EC2          = pio4!out3
E_EC3          = pio4!out4
N14CR0805170   = pio4!out5

N13HCPL31500   = pio4!in1
N13HCPM31500   = pio4!in2
N13HCPN31500   = pio4!in3
N13HCPO31500   = pio4!in4

bus0           = pio4!out20
bus1           = pio4!out21
bus2           = pio4!out22
bus3           = pio4!out23
bus4           = pio4!out24
bus5           = pio4!out25
bus6           = pio4!out26
bus7           = pio4!out27
```

Like all GTSL libraries, the DIO manager is configured in an `application.ini` file. The keyword `DIODevice` refers to the R&S TS-PIO4 module in the `physical.ini`. If more than one digital module is to be used, a DIODevice2, DIODevice3, and so on is added.

The "DIO channel table" refers to the associated table in the file.

The left side of the channel table contains the logical names and the right side the link to the physical channels on the individual modules.

Channel names for groups (e.g. buses) do not have any brackets. Here they are called "bus0" for example, whereas they are called "bus[0]" in the waveform file.

Channel names are not case-sensitive.

5.2.2.3 Structure of a Test Program

Table 5-4: These functions must be called once at beginning of test program.

Function	Comments
`RESMGR_Setup`	Call of the resource manager
`DIOMGR_Setup`	Call of the DIO manager and initialization of the TS-PDFT module
`DIOMGR_ConfigureStimulus` `DIOMGR_ConfigureResponse`	Configuration of logical levels for inputs and outputs
`DIOMGR_LoadWaveform`	Loading a pattern set from a file to the memory.
`DIOMGR_ConfigurePatternSetTiming`	Definition of the timing of a pattern set

Table 5-5: These functions are typically called in a loop over multiple UUTs.

Function	Comments
`DIOMGR_ExecutePatternSet`	Execution of a pattern set and return of the result (pass/fail)
Optional functions:	Generally only called if the pattern set is defective (fail)
`DIOMGR_SaveWaveform`	Storage of the results as a TBL file
`DIOMGR_GetPatternSetExecutedPatternCount`	Reading out of the executed patterns
`DIOMGR_GetPatternSetFailedPatternCount`	Reading out of the failed patterns
`DIOMGR_GetPatternSetFailedChannelCount`	Reading out of the failed channels
`DIOMGR_GetPatternSetFailedChannelNames`	Reading out of failed channels as a list of names (comma-separated)
`DIOMGR_GetPatternSetChannelData`	Reading out of the current response message (measured data) for a channel
`DIOMGR_GetPatternSetChannelResults`	Reading out of the results for a channel (pass/fail)

Table 5-6: After all tests have been performed, these functions are used for cleaning up.

Function	Comments
`DIOMGR_Cleanup`	Closes the DIO manager
`RESMGR_Cleanup`	Closes the resource manager

5.2.2.4 Ports

The R&S TS-PIO4 module is divided into 8 ports each with 4 channels. Each port can be set to different driver and sensor levels. If different levels are used in a program, `DIOMGR_ConfigureStimulus()` and `DIOMGR_ConfigureResponse()` must be called once for each "logic family" and the required level must be set.

5.2.2.5 Loading of Waveform File

The waveform file defines a pattern set with specification of the timestamps. These timestamps do not have to be equidistant. Normally, a new line is created whenever at least one signal changes.

```
PATTERN
    0.0> 0 0 0 0 0 00 = X X X X
 1000.0> 1 0 0 0 0 01 = X X X X
 1001.0> 1 1 0 0 0 01 = X X X X
 1050.3> 1 1 1 0 1 01 = 1 1 1 1
 2100.0> Z Z Z Z Z FF = X X X X
etc.
```

When the waveform is loaded to the R&S TS-PIO4 module, the timing must be converted to a fixed framework which is determined by the pattern set period of the module. The loading function uses the "timeGrid" parameter to generate the samples of the waveform in a fixed time grid. Each sample is then converted into a pattern and stored in the memory of the module.

If a "timeGrid" of 500 ns is set in the example above, this would result in the following samples:

Pattern	Time	Inputs	Outputs
1	0 ns	0 0 0 0 0 00	X X X X
2	500 ns	0 0 0 0 0 00	X X X X
3	1000 ns	1 0 0 0 0 01	X X X X
4	1500 ns	1 1 1 0 1 01	1 1 1 1
5	2000 ns	1 1 1 0 1 01	1 1 1 1
6	2500 ns	Z Z Z Z Z FF	X X X X

Some patterns are repeated (1 and 2, 4 and 5) and some are not recognized as they are too short and are between the sampling time points (e.g. patterns at the timestamp 1001.0).

The timestamp and the "timeGrid" parameter do not necessarily have to represent the actual execution speed. The actual execution speed is set using the `rspio4_ConfigurePatternSetTiming()` function. This also means that the waveform file does not need to be modified if the test is to run at a different pattern rate.

The following points should be observed when writing the waveform file:

- Use equidistant timestamps
- Timestamps are to reflect the real timing, i.e. exactly the timing that is used for the test.
- Use the interval between the timestamps as the value for the "timeGrid" parameter.
- Use the interval between the timestamps as the "Pattern Set Period".

5.2.2.6 Execution of Pattern Set

The pattern set can be executed synchronously or asynchronously. In the first case, the `DIOMGR_ExecutePatternSet()` function does not return until execution has been completed (or the maximum time has been exceeded). In the second case, execution is started or the trigger armed and the function returns without having to wait for actual execution of the pattern set. The `DIOMGR_WaitUntilPatternSetComplete()` function can be used to wait for the end of execution.

5.3 Configuration of Digital Channels

The module offers a series of configurable parameters. This section describes the configuration options and lists the corresponding driver functions. For details on the driver functions, refer to the GTSL help of the driver software.

5.3.1 Setting of Voltage Range

The application must set a range that contains the desired voltages for the high and low levels of the outputs as well as the required comparator thresholds for the inputs. Each port can be assigned its own voltage range.

Switchover of the voltage ranges causes long settling times on the module which have to be taken into consideration in the device driver. When the voltage range is adjusted, all drivers are switched to high impedance.

The voltage ranges are listed in the Voltage ranges table.

- `rspio4_ConfigurePortVoltageRange`

5.3.2 Configuration of Stimulus Channels

Settings for the output voltages can be programmed for each port.

- `rspio4_ConfigureStimPort`

The settings are made for one or more ports depending on the parameters in the function call. The current limit always applies to the total output currents from all 4 drivers of a port.

When current is flowing, a voltage drop that may have to be taken into consideration during level adjustment is caused as a result of the output impedance of the driver and the resistors in the output safety circuits (together approx. 30 Ω).

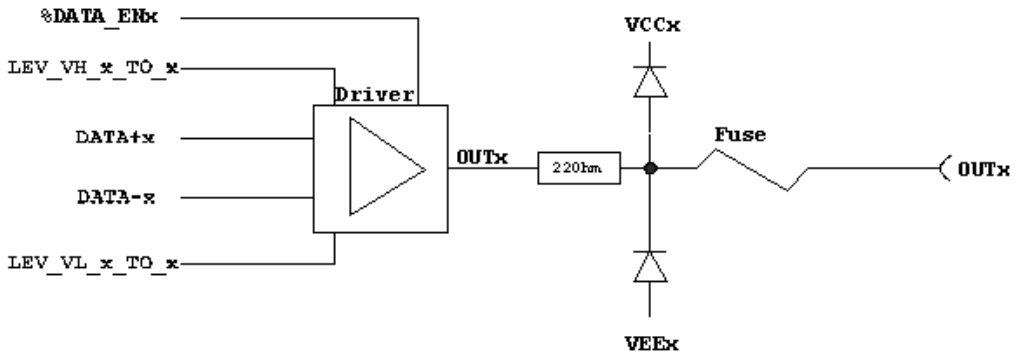


Figure 5-2: Configuration of stimulus channels

5.3.3 Configuration of Input Channels

- `rspio4_ConfigureRespPort`

The measurement channel inputs have a safety circuit upstream of the comparators. This safety circuit has the following structure:

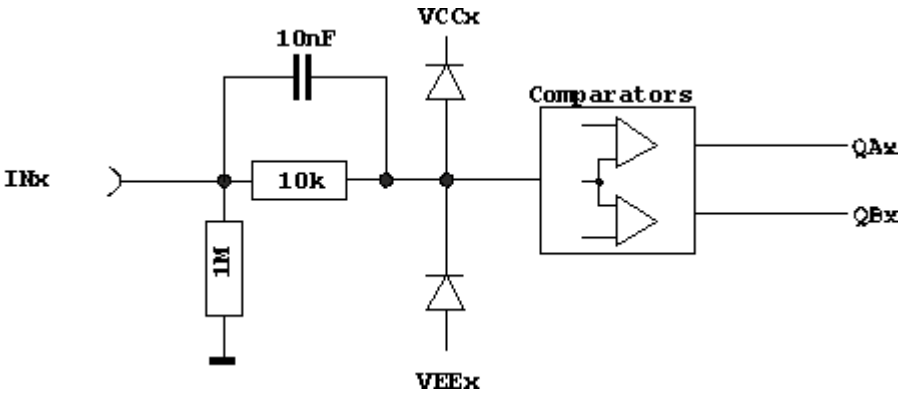


Figure 5-3: Configuration of input channels

Each input is routed to two comparators which have an adjustable threshold. This makes it possible to implement a hysteresis for evaluation of the signals. The thresholds can be set using the driver function `rspio4_ConfigureRespPort`. This allows individual values to be set for each port.

The result of the signal evaluation of a channel is **1** if the input level is greater than the threshold for the high level.

The result of the signal evaluation of a channel is **0** if the input level is less than the threshold for the low level.

If the input level is between the thresholds, "Forbidden Zone" is returned as the result. The states of the two comparators can be read out using the low-level driver function `rspio4_GetDinHighAndLowComp`.

If the comparator is set to "COMP" mode, the two comparators function as window comparators. A **1** is detected if the input level is between the thresholds, and a **0** is detected if the input level is below the threshold for low or above the threshold for high.

5.3.4 Time Settings for Data Output

This setting is required both for dynamic tests in compliance with IVI Digital and for dynamic tests with the low-level driver functions.

To configure the time response of the stimulus channels during output of a pattern set, the `rspio4_ConfigureStimTiming` function is used to set the trigger delay, the pattern duration and the number of patterns to be output (Trigger Delay, Pattern Period, Loop Count). The trigger delay is the wait time between the trigger event and output of the first pattern. The pattern duration is the time during which an individual pattern is present.

5.3.5 Time Settings for Data Recording

This setting is required both for dynamic tests in compliance with IVI Digital and for dynamic tests with the low-level driver functions.

The time parameters required for recording the input signals can be set using the `rspio4_ConfigureRespTiming` function. As with data output, the trigger delay, the pattern duration as well as the number of patterns to be read in (Response Trigger Delay, Response Pattern Period, Loop Count) are available as parameters.

The trigger delay determines the wait time between the trigger event and initial sampling. The response trigger delay is set such that the UUT data is available at the input in a stable condition at this point in time. In the typical case where the Stimulus Pattern Period and Response Pattern Period are identical, a delay of 0.0 meant that in every pattern sampling would take place directly at the beginning of the pattern; a delay greater than the pattern period would mean sampling within the next pattern or within a different pattern.

Example:

Stimulus pattern period: 50 ns

Response pattern period: 50 ns

Stim. trigger delay: 0.0 ns

Resp. trigger delay: 25.0 ns

The outputs are changed at time point 0. The data is sampled 25.0 ns later, i.e. at the middle of the pattern in our example. Here, it is therefore assumed that the UUT outputs stable data after less than 25.0 ns.

5.3.6 Configuration of Data Width in Dynamic Mode

- `rspio4_ConfigureStimMode`

- `rspio4_ConfigureRespMode`

If required, the digital channels can be configured in different combinations for simultaneous static and dynamic operation. This, for example, allows the configuration of control signals that remain at a constant level during entire test sequences. They then no longer have to be part of the dynamic data. This also simplifies the creation of pattern sets.

This type of configuration is only possible if programming is performed using the low-level driver functions. If the driver functions compliant with IVI Digital are used, the dynamic width is always 32 bits.

The possible or selected constellations must, however, be taken into consideration during fixture wiring.

The following configurations are possible (see also `rspio4.h`).

Mode	Configuration
`RSPIO4_VAL_CTRL_STATIC`	Static mode
`RSPIO4_VAL_CTRL_4BIT`	4 dynamic DOUT or DIN: OUT 1 to 4 / IN 1 to 4 No tri-state
`RSPIO4_VAL_CTRL_8BIT`	8 dynamic DOUT or DIN: OUT 1 to 8 / IN 1 to 8 No tri-state
`RSPIO4_VAL_CTRL_16BIT`	16 dynamic DOUT or DIN: OUT 1 to 16 / IN 1 to 16 No tri-state
`RSPIO4_VAL_CTRL_32BIT`	32 dynamic DOUT or DIN: OUT 1 to 32 / IN 1 to 32 No tri-state
`RSPIO4_VAL_CTRL_4BIT_TRISTATE`	4 dynamic DOUT or DIN: OUT 1 to 4 / IN 1 to 4 Tri-state information in data
`RSPIO4_VAL_CTRL_8BIT_TRISTATE`	8 dynamic DOUT or DIN: OUT 1 to 8 / IN 1 to 8 Tri-state information in data
`RSPIO4_VAL_CTRL_16BIT_TRISTATE`	16 dynamic DOUT or DIN: OUT 1 to 16 / IN 1 to 16 Tri-state information in data
`RSPIO4_VAL_CTRL_32BIT_TRISTATE`	32 dynamic DOUT or DIN: OUT 1 to 32 / IN 1 to 32 Tri-state information in data

The `RSPIO4_VAL_CTRL_32BIT_TRISTATE` mode is always used in IVI-Digital-compliant operation.

5.3.7 Power Consumption

5.3.7.1 Estimation of Power Consumption

The total power consumption of the R&S TS-PIO module should not exceed 40 W and depends on the operating mode of the input and output channels.

The following description explains calculation of the maximum power consumption for the operating ranges most frequently used.

VH = Output level High

VL = Output level Low

Range 3 with the settings VL = +1.5 V to +6.0 V and VH = +1.5 V to +10.0 V

Range 2 with the settings VL = –1.8 V to +2.7 V and VH = –1.8 V to +10.0 V

Range 1 with the settings VL = –3.5 V to +1.0 V and VH = –3.5 V to +10.0 V

Range 0 with the settings VL = –6.0 V to –1.5 V and VH = –6.0 V to +8.0 V

Output channels

Table 5-7: Maximum power consumption per group (consisting of 4 channels)

Range	Power consumption in a group [W]	Maximum power in a group with four channels
3	#CH * 0.021 * f[MHz] * (0.421 + 0.233 * U_swing)	< 3.9 W
2	#CH * 0.036 * f[MHz] * (0.548 + 0.172 * U_swing)	< 3.9 W
1	#CH * 0.047 * f[MHz] * (0.605 + 0.143 * U_swing)	< 3.9 W
0	#CH * 0.051 * f[MHz] * (0.609 + 0.137 * U_swing)	< 3.9 W

#CH: Number of active channels in a group with four channels

f: IO switchover frequency (= 1/2 sample rate), duty cycle 50%, MHz

U_swing : Voltage variation of the IO signal (VH-VL), V

Example:

Four active channels in a group with a switchover frequency of 20 MHz (corresponds to a sample rate of 40 Msamples), VL = +1.5 V and VH = +4.5 V.

- U_swing = VH – VL = 4.5 V – 1.5 V = **3.0 V**
- The values for VL and VH can be used in the ranges 2 and 3

Calculation of the power consumption in a group:

- *Range 3:* 4 * 0.021 * 20 * (0.421 + 0.233 * 3.0) = 1.88 W => **OK** (< 3.9 W)
- *Range 2:* 4 * 0.036 * 20 * (0.548 + 0.172 * 3.0) = 3.06 W => **OK** (< 3.9 W)

Range 3 must be used on account of the limited power consumption of the group.

Table 5-8: Maximum power consumption per module

Range	Power consumption of active IO channels [W]	Maximum power consumption of a module [W]
3	#GRP * #CH * 0.021 * f[MHz] * (0.421 + 0.233 * U_swing) + ∑P_load	< 25 W
2	#GRP * #CH * 0.036 * f[MHz] * (0.548 + 0.172 * U_swing) + ∑P_load	< 22 W
1	#GRP * #CH * 0.047 * f[MHz] * (0.605 + 0.143 * U_swing) + ∑P_load	< 20 W
0	#GRP * #CH * 0.051 * f[MHz] * (0.609 + 0.137 * U_swing) + ∑P_load	< 19 W

#GRP: Number of active groups (a group consists of max. four channels), max. 8 groups

#CH: Number of active channels in a group with four channels

f: IO switchover frequency (= 1/2 sample rate), duty cycle 50%, MHz

U_swing: Voltage variation of the IO signal (VH-VL), V

Example:

Four active channels and 8 active groups with a switchover frequency of 20 MHz (corresponds to a sample rate of 40 Msamples), VL = +1.5 V and VH = +4.5 V.

- U_swing = VH – VL = 4.5 V – 1.5 V = **3.0 V**
- The values for VL and VH can be used in the ranges 2 and 3

Calculation of the power consumption for one module:

- *Range 3:* 8 * 4 * 0.021 * 20 * (0.421 + 0.233 * 3.0) = 15.05 W => **OK** (< 25 W)
- *Range 2:* 8 * 4 * 0.036 * 20 * (0.548 + 0.172 * 3.0) = 24.51 W => **N.OK** (> 22 W)

In this case, only range 3 can be used.

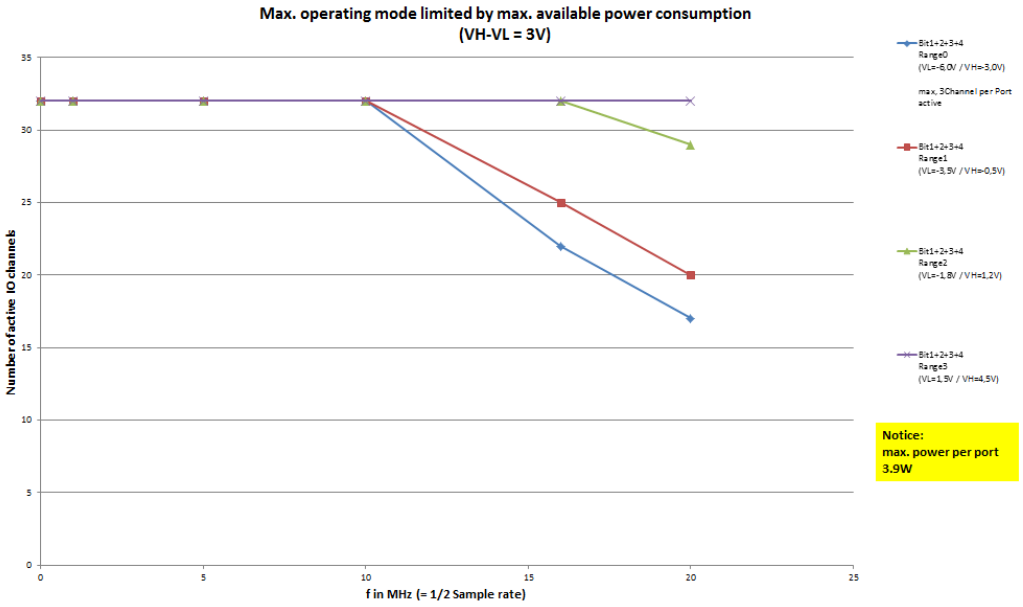


Figure 5-4: Max. operating range as a function of maximum power consumption

Input channels

Depending on the operating range and frequency, receive channels (comparators) are additionally subject to dissipation loss during switchover.

The table below gives the approximate additional power consumption for the operating ranges with the following output values:

- Max. switchover frequency: 20 MHz
- Voltage variation: 3.3 V
- Number of channels: 32 input channels

Range	Additional power consumption (typ.)
3	10.0 W
2	5.5 W
1	12.5 W
0	13.5 W

Total power consumption

The input and output channels must be taken into consideration when estimating the total power consumption.

5.3.7.2 Safety Mechanism

If the maximum permissible power consumption is exceeded, the IOs are deactivated by a safety circuit. In this case, an error will occur when a driver function is called. The IOs can be reactivated with the `rspio4_reset` function.

5.4 Triggering and Sequence Control

5.4.1 Trigger Units

There are two trigger units on the module. The trigger units control output of stimulus data as well as reading in of the data at the digital inputs. Trigger unit IT1 is responsible for outputs of the stimulus data, and trigger unit IT2 is responsible for reading in of the response data.

The trigger units can be configured for various input conditions. It is also possible to configure which output signal they are to generate and where this signal is to be routed to.

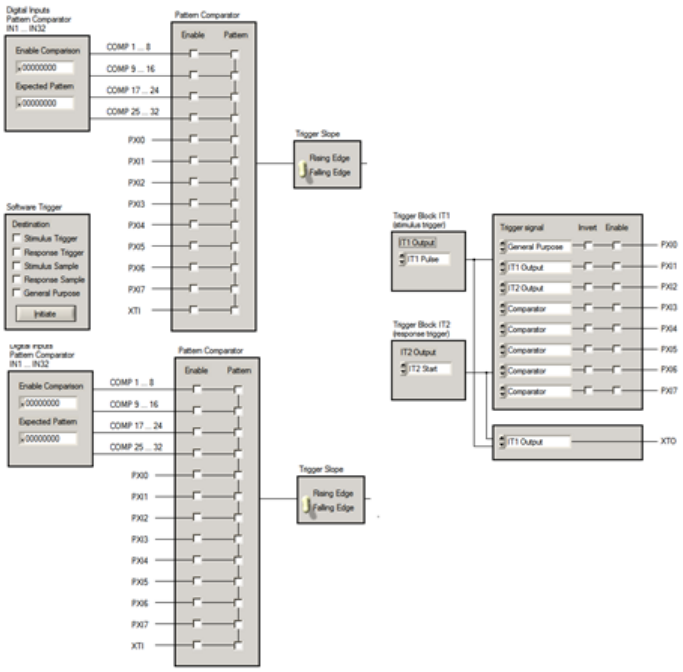


Figure 5-5: Structure of trigger units for stimulus and response

Inputs of trigger unit:

On the input side, the `rspio4_ConfigHWTriggerInput` function is used to configure whether and which inputs (PXI0 to PXI7, XTI, 4 outputs of the digital inputs pattern comparator) are to be used as a hardware trigger and with which logical state, and which edge is used.

The `rspdft_ConfigDinComparator` function allows setup of the digital inputs pattern comparator.

Alternatively, the unit can also be triggered using a software trigger (e.g. the `rspio4_executePattern` function starts the processing of a pattern set by means of a software trigger by calling the `rspio4_InitiateSWTrigger` function).

If the trigger unit was started by an event, then as many stimulus and response patterns are processed as were set using the configuration functions `rspio4_ConfigureStimTiming` and `rspio4_ConfigureRespTiming` in the `Loop Count` parameters.

Outputs of trigger unit:

The output of the trigger unit is used to actuate other functions. Selection of a certain type of output signal does not influence the internal function (output stimulus, read in response).

This output signal (e.g. ACTIVE) can be output on the PXI0 to 7 lines as well as on XTO in order to, for example, trigger a measurement on a different CompactPCI board.

5.4.2 Receiving of Trigger Signals

Various trigger sources are available for starting the controllers:

Designation	Remarks
SW Trigger	Sequence control starts immediately after arming (when the `rspio4_InitiateSWTrigger` function is called). This is the default sequence if a pattern set is to be processed under program control. For configuration, see also the example for the dynamic test with low-level functions.
"XTI"	TTL input on the front connector X10; a positive or negative edge starts the sequence. Triggering via XTI is configured with the `rspio4_ConfigHWTriggerInput` function. It is used to define the edge and to activate/deactivate triggering via XTI.
PXI0 to PXI7	Positive or negative edges on these lines start the sequence. The active edge and which trigger signal is used are again set using the `rspio4_ConfigHWTriggerInput` function. If multiple trigger sources are selected, then this is equivalent to an OR logic operation.
Pattern Comparator	A pattern comparator which evaluates the states of the 32 inputs (IN) can also be used as the trigger source. This is again also configured using `rspio4_ConfigHWTriggerInput` as well as `rspio4_ConfigDinComparator` in order to define the condition for the input pattern comparator.

The `rspio4_ConfigHWTriggerInput` function determines the trigger source. Afterwards the hardware trigger sources are armed using the `rspio4_EnableHWTrigger`

function and the addressed sequence controls are in the "Waiting" state. If a software trigger has been triggered, the associated control immediately switches to the "Running" state. Otherwise the state change does not take place until the trigger event has occurred. In this state, the number of patterns in the stimulus memory (Stim Loop Count) is then output and the number of patterns (Resp Loop Count) in the reference memory is recorded.

If only patterns are to be output, the `rspio4_InitiateSWTrigger` function has to be called with the mask RSPIO4_IT_MASK_IT1 only. If only recording is required, the function must be called using only RSPIO4_IT_MASK_IT2.

After the available number of patterns has been processed, the associated sequence control switches back to the "Stopped" state.

The current state of both sequence controls can be queried using the `rspio4_GetItStatus` function. By calling the `rspio4_WaitUntilPatternSetComplete` function, the sequence control can be made to wait until the end of execution in the test program.

If sequence control is in the "Waiting" or "Running" state, some driver functions cannot be performed. In this case, these functions return an error message. If necessary, the `rspio4_AbortExecution` function can be used to switch sequence control to the default state.

5.4.3 Generation of Trigger Signals

The R&S TS-PIO4 module is able to generate trigger signals on the following lines:

Designation	Remarks
XTO	TTL output on the front connector
PXI0 to PXI7	PXI trigger lines on the backplane

For a change to occur on the trigger lines, an event which provokes the trigger pulse must be assigned to the selected line.

The following signals can be routed to the output using the `rspio4_ConfigXTO` function:

- Output of pattern comparator
- Output of IT1
- Output of IT2

The following signals can be switched to a selected PXI using the `rspio4_ConfigPxiTrigOut` function:

- Output of pattern comparator
- Output of IT1
- Output of IT2
- Output of general purpose trigger generator

This function can also be used to invert the selected signal and to activate the output driver of the PXI trigger line.

The `rspio4_InitiateSWTrigger` function is used to start the following function blocks under software control:

- Trigger logic block IT1 (sequence control for data output)
- Trigger logic block IT2 (sequence control for data recording)
- Sample trigger stimulus (output of a single pattern from the stimulus memory)
- Sample trigger response (recording of a single pattern to the response memory)
- General purpose trigger generator (generation of a single trigger pulse)

The `rspio4_ConfigItTrigOut` function can be used to determine which signal the trigger logic blocks IT1 and IT2 are to output after they have been started either via software (`rspio4_InitiateSWTrigger`) or by means of a configured hardware signal (`rspio4_ConfigHWTriggerInput`). The table below shows the options:

Designation	Remarks
ITn ACTIVE	Pulse while the pattern set is being output.
ITn OUT	Trigger block outputs a pulse for each pattern. The pulses are relatively short (minimum pattern rate/2).
ITn STATUS	Signals the internal state of the trigger block. The signal becomes active when the trigger event is received. The signal does not become inactive again until the trigger block is reset.
ITn START	A pulse after the trigger condition has been detected.

See also the example in Chapter 7.3.6, "Triggered Pattern Execution", on page 73.

5.5 PWM

The module supports the output of PWM signals at any output (OUTx).

PWM is configured using the `rspio4_ConfigurePWM` function. The frequency is set in Hz and the duty cycle is specified in %.

The `rspio4_SetStatePWM` function is used to switch PWM on/off for individual channels.

The time resolution is 12.5 ns if the internal reference clock of 80 MHz is used. Lower frequencies are generated using a divider. With this method, there are certain limitations with regard to the choice of frequency and duty cycle.

The following examples show the correlations at the highest frequencies:

- 40 MHz: 0 %, 50 %, 100 %
- 26.7 MHz: 0 %, 33 %, 66 %, 100 %
- 20 MHz: 0 %, 25 %, 50 %, 75 %, 100 %

5.6 Frequency Measurement

The R&S TS-PIO4 module supports frequency measurement at inputs IN1 to IN32. The `rspio4_FrequencyMeasurement` function is used for this purpose. A distinction is made between two methods, depending on the passed parameters:

- Specification of a gate time: Pulses are counted until the gate time has expired. This allows the frequency value to be calculated. The "gateCount" parameter is set to 0.
- If a gate time of 0 and a defined number of pulses (gateCounts) are specified, the module measures the time required for a certain number of pulses to arrive. The value is also converted into a frequency.

5.7 Bidirectional Channels

Each input can be individually connected to the associated output by means of an analog switch. This allows reading out of the outputs and reading in of a UUT response with the outputs switched off (tri-state). The connection is set up using the `rspio4_ConnectInOut` function.

5.8 External Clock Input

Various applications can be implemented using the external clock input (EXT_CLK). One possibility is to generate output frequencies that cannot be achieved using an internal frequency divider; alternatively, the measuring card can be synchronized to the clock of a UUT. The minimum permissible input frequency is 20 MHz and the maximum is 40 MHz. All lower frequencies can be generated using the internal divider.

The clock signal source (internal PXI 10 MHz clock or supplied via EXT_CLK) can be selected using the `rspio4_ConfigureClock` function. If an external clock is selected, the clock frequency must be specified using the parameter `ExternClockFreq`. This parameter is ignored if the internal clock is selected. The state of the clock PLL can be queried using the `rspio4_GetPllLockedStatus` function. This function indicates whether the clock can be used successfully.

The changed clock changes the timebase of the card. This increases, for example, the step size in the case of adjustable trigger delays.

5.9 Synchronization of Multiple Modules

Multiple modules can output and record data in a synchronized manner via the various trigger inputs and outputs. The internal PXI trigger lines are not length-compensated which means that significant delay differences can occur. The highest level of accuracy

can be achieved by means of length-compensated lines between the master's XTO and the XTIs.

5.10 AUX Channels

Each of the AUX1, AUX2, AUX3 and AUX4 signals are routed from the rear X20 connector directly to the front X10 connector. The current-carrying capacity is 1 A in each case.

5.11 GND Relay

The GND relay connects the GNDNO signals to GND at the front X10 connector. This means that a UUT can optionally be connected to GND. With in-circuit tests, the UUT must be floating, whereas with functional tests it is often recommended to have a connection between UUT ground and system ground.

In the default state, this connection is open.

6 Startup

6.1 Installation of R&S TS-PIO4 Module

To install the R&S TS-PIO4 plug-in module, proceed as follows:

NOTICE

Damage to backplane caused by bent pins

Bent pins can cause permanent damage to the backplane.

Check the backplane connectors for bent pins.

Any bent pins must be straightened.

When inserting the plug-in module, guide it with both hands and carefully push it into the backplane connectors.

1. Power down and switch off the R&S CompactTSVP.
2. Select a suitable front-side slot.
3. Remove the corresponding section of the front panel from the TSVP chassis by undoing the screws.
4. Push in the plug-in module using moderate pressure.
5. The upper catch pin of the plug-in module must be inserted into the right hole in the TSVP chassis and the lower catch pin into the left hole.

   The module is correctly located when a distinct "stop" can be felt.
6. Securely tighten the screws at the top and bottom of the front panel of the plug-in module.

7 Software

7.1 Driver Software

A LabWindows IVI driver is available for actuation of the R&S TS-PIO4 digital functional test module. All additional functions of the hardware are controlled using specific extensions of the driver. The driver is part of the ROHDE & SCHWARZ GTSL software. All the functions of the driver are described fully in the online help and in the LabWindows/CVI function panels. The following software modules are installed during driver installation:

Module	Path	Remarks
`rspio4.dll`	`<GTSL directory>\Bin`	Driver
`rspio4.chm`	`<GTSL directory>\Bin`	Help file
`rspio4.fp`	`<GTSL directory>\Bin`	LabWindows/CVI function panel file, function panels for CVI development environment
`rspio4.sub`	`<GTSL directory>\Bin`	LabWindows/CVI attribute file. This file is required by some "function panels".
`rspio4.lib`	`<GTSL directory>\Bin`	Import library
`rspio4.h`	`<GTSL directory>\Include`	Header file for driver

The IVI and VISA libraries from National Instruments are needed to run the driver.

7.2 Soft Panel

The software package of the R&S TS-PIO4 module includes a soft panel (see Figure 7-1). The soft panel is based on the IVI driver and enables interactive operation of the module.

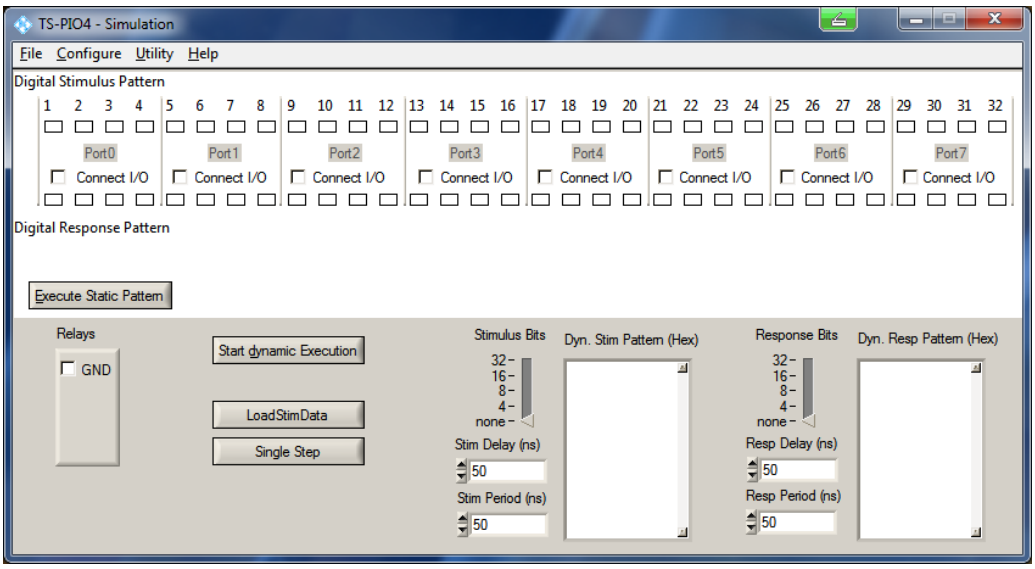


Figure 7-1: Soft panel of R&S TS-PIO4

Operation of the soft panels is described in *Software Description for R&S GTSL*.

In the R&S TS-PIO4 soft panel, it is possible to set static and dynamic bit patterns at the outputs and to read in input states.

The voltage ranges, the high and low driver levels as well as threshold voltages of the input comparators can be set in subsequent dialogs which are opened from the Configure menu.

Furthermore, it is also possible to configure the stimulus and response trigger units as well as to provoke trigger events and to make PWM settings.

7.3 Programming Examples for R&S TS-PIO4

7.3.1 Digital Test with "DIO Manager" Library

This example shows the execution of a digital test using the GTSL library "DIO Manager" (DIOMGR). The input file (`823916.tbl`) and the associated configuration file (`pio4Application.ini`) are described in Chapter 5.2.2, "Digital Test with DIO Manager", on page 24. Additional information on the structure of the configuration files (physical layer configuration file and application layer configuration file) and on the function of the resource manager (RESMGR) can be found in the manual *Software Description R&S GTSL*.

7.3.1.1 Main Function

```
/* Programming example with DIOMGR */
#include <ansi_c.h>
#include "resmgr.h"
#include "diomgr.h"

static short errorOccurred;
static long  errorCode;
static char  errorMessage[GTSL_ERROR_BUFFER_SIZE];
static long  residDiomgr;

static char benchName[]      = "bench->823916";
static char fileName[]       = "823916.tbl";
static char fileNameResult[] = "823916result.tbl";

/* list of stimulus channels */
static char stimChannels[] = "N14CR0805170,E_COM,E_EC1,E_EC2,E_EC3";
static char respChannels[] = "N13HCPN31500,N13HCPM31500,"
                             "N13HCPO31500,N13HCPL31500";

/* prototypes */
static void cs ( char * funcName );
static void runTest ( void );
static void diagnosis ( void );

/* FUNCTION ******************************************************************/
/* loads the libraries and runs the test
*******************************************************************************/
int main (int argc, char *argv[])
{
  printf("Example using DIOMGR functions for dynamic pattern execution\n\n");

  /* setup libraries */
  RESMGR_Setup (0, "physical.ini", "pio4Application.ini",
    &errorOccurred, &errorCode, errorMessage);
  cs("RESMGR_Setup");

  if ( ! errorOccurred )
  {
    DIOMGR_Setup (0, benchName, &residDiomgr,
      &errorOccurred, &errorCode, errorMessage);
    cs("DIOMGR_Setup");
  }

  if ( ! errorOccurred )
  {
    runTest ( );
  }
```

```
  /* cleanup libraries */
  DIOMGR_Cleanup (0, residDiomgr, &errorOccurred, &errorCode, errorMessage);
  cs("DIOMGR_Cleanup");

  RESMGR_Cleanup ( 0, &errorOccurred, &errorCode, errorMessage);
  cs("RESMGR_Cleanup");

  printf("\nPress 'Enter' to terminate\n");
  getchar();

  return 0;
}
```

7.3.1.2 Error Handling

This function checks the return values of a function from the GTSL libraries (RESMGR, DIOMGR). A message is issued if there is an error.

```
/* FUNCTION ******************************************************************/
/* checks the return status of a library call
*******************************************************************************/
static void cs ( char * funcName )
{
  if ( errorOccurred )
  {
    printf ("%s returned 0x%08X\n%s\n\n", funcName, errorCode, errorMessage);
  }
}
```

7.3.1.3 Execution of Digital Test

```
/* FUNCTION ******************************************************************/
/* Configures the module for digital test, loads the test and executes it.
   The results are uploaded from the module and stored as a result TBL file.
*******************************************************************************/
static void runTest ( void )
{
  long numPatterns;
  long patternSetResult;
  char usedChannels[512];

  /* configure voltage ranges */
  strcpy(usedChannels, stimChannels);
  strcat(usedChannels, ",");
  strcat(usedChannels, respChannels);

  DIOMGR_ConfigureVoltageRange (0, residDiomgr, usedChannels, "VOLTAGE_RANGE_2",
    &errorOccurred, &errorCode, errorMessage);
  cs("DIOMGR_ConfigureVoltageRange");
```

```
  /* configure stimulus and response levels */
  DIOMGR_ConfigureStimulus (0, residDiomgr, stimChannels, "TTL", 3.3, 0.1,
    &errorOccurred, &errorCode, errorMessage);
  cs("DIOMGR_ConfigureStimulus");

  DIOMGR_ConfigureResponse (0, residDiomgr, respChannels, "HYSTERESIS",
    0.8, 2.0, &errorOccurred, &errorCode, errorMessage);
  cs("DIOMGR_ConfigureResponse");

  /* load the waveform. Pattern set name = file name */
  DIOMGR_LoadWaveform (0, residDiomgr, fileName, "TBL", 1.0e-6, fileName,
    &numPatterns, &errorOccurred, &errorCode, errorMessage);
  cs("DIOMGR_LoadWaveform");

  /* configure the timing */
  DIOMGR_ConfigurePatternSetTiming (0, residDiomgr, fileName, 1.0e-6,
    0.5e-6, 1.0, &errorOccurred, &errorCode, errorMessage);
  cs("DIOMGR_ConfigurePatternSetTiming");

  /* run the test */
  DIOMGR_ExecutePatternSet (0, residDiomgr, fileName, "SYNCHRONOUS",
    &patternSetResult, &errorOccurred, &errorCode, errorMessage);
  cs("DIOMGR_ExecutePatternSet");

  if ( DIOMGR_VAL_RESULT_FAILED == patternSetResult )
  {
    printf ( "Pattern set failed\n" );
    diagnosis ();
  }
  else
  {
    printf ( "Pattern set passed\n" );
  }

  DIOMGR_SaveWaveform (0, residDiomgr, fileName, fileNameResult, "TBL",
    &errorOccurred, &errorCode, errorMessage);
  cs("DIOMGR_SaveWaveform");

  DIOMGR_UnloadWaveform (0, residDiomgr, fileName,
    &errorOccurred, &errorCode, errorMessage);
  cs("DIOMGR_UnloadWaveform");
}
```

7.3.1.4 Evaluation of Failed Patterns

```
/* FUNCTION ****************************************************************/
/* reads information about pattern set failures and prints it to stdout
```

```
*****************************************************************************/
static void diagnosis ( void )
{
  long executedPatternCount;
  long failedPatternCount;
  long failedChannelCount;
  char failedChannelNames[1024];
  long bufferSize;
  char * pData = NULL;
  char * pResults = NULL;
  char * token = NULL;
  int i;
  int numFail;

  /* read results about pattern set failure */
  DIOMGR_GetPatternSetExecutedPatternCount (0, residDiomgr, fileName,
    &executedPatternCount, &errorOccurred, &errorCode, errorMessage );
  cs("DIOMGR_GetPatternSetExecutedPatternCount");
  printf ( "%d patterns executed\n", executedPatternCount );

  DIOMGR_GetPatternSetFailedPatternCount (0, residDiomgr, fileName,
    &failedPatternCount, &errorOccurred, &errorCode, errorMessage );
  cs("DIOMGR_GetPatternSetFailedPatternCount");
  printf ( "%d patterns failed\n", failedPatternCount );

  DIOMGR_GetPatternSetFailedChannelCount (0, residDiomgr, fileName,
    &failedChannelCount, &errorOccurred, &errorCode, errorMessage );
  cs("DIOMGR_GetPatternSetFailedChannelCount");
  printf ( "%d channels failed:\n", failedChannelCount );

  DIOMGR_GetPatternSetFailedChannelNames (0, residDiomgr, fileName,
    sizeof(failedChannelNames), failedChannelNames,
    &errorOccurred, &errorCode, errorMessage );
  cs("DIOMGR_GetPatternSetFailedChannelNames");
  printf ( "   %s\n", failedChannelNames );

  /* allocate memory for data and pass/fail */
  bufferSize = executedPatternCount + 1;
  pData = malloc ( bufferSize );
  pResults = malloc ( bufferSize );

  /* cycle thru the channel names, output data and mark errors */
  token = strtok (failedChannelNames, ",");
  while ( token )
  {
    /* "token" contains a failed channel name */
    DIOMGR_GetPatternSetChannelData ( 0, residDiomgr, fileName, token,
      bufferSize, pData, &errorOccurred, &errorCode, errorMessage );
    cs("DIOMGR_GetPatternSetChannelData");
```

```
    DIOMGR_GetPatternSetChannelResults ( 0, residDiomgr, fileName, token,
      bufferSize, pResults, &errorOccurred, &errorCode, errorMessage );
    cs("DIOMGR_GetPatternSetChannelResults");

    /* 1=passed, 0=failed. Replace "failed" by an X and "passed" by a space */
    numFail = 0;
    for ( i=0; i<bufferSize; i++ )
    {
      switch ( pResults[i] )
      {
        case '0':
          pResults[i] = 'X';
          numFail++;
          break;
        case '1':
          pResults[i] = ' ';
          break;
        default:
          break;
      }
    }

    printf ( "\n%s : %d fails\n", token, numFail );
    printf ( "- Data    : %s\n", pData );
    printf ( "- Results : %s\n", pResults );

    /* get next channel name */
    token = strtok ( NULL, "," );
  }


  free ( pData );
  free ( pResults );
}
```

7.3.2 Dynamic Pattern Execution with IVI Digital

7.3.2.1 Main Function

The return value of a device driver call is stored in the module-global variable `sta`. The status code is checked using the `chk()` function.

```
/* Example using IVI functions for dynamic pattern execution */
#include <ansi_c.h>
#include "rspio4.h"

/* adapt the resource descriptor to your test system! */
static char resDesc[] = "PXI5::11::INSTR";
```

```
static char patternSetName[] = "823916";

/* channel names */
static char o1[] = "OUT1";
static char o2[] = "OUT2";
static char o3[] = "OUT3";
static char o4[] = "OUT4";
static char o5[] = "OUT5";

static char bus[] = "OUT20-OUT27";
static char out[] = "OUT1-OUT31";

static char i1[] = "IN1";
static char i2[] = "IN2";
static char i3[] = "IN3";
static char i4[] = "IN4";

static char in[] = "IN1-IN32";

static ViStatus sta;
static ViSession vi;

/* prototypes */
static void chk ( char * funcName );
static void runTest ( void );
static void createPatternSet ( void );
static void diagnosis ( void );

/* FUNCTION ******************************************************************/
/* loads the driver and runs the test
*******************************************************************************/
int main (int argc, char *argv[])
{
  printf("Example using IVI functions for dynamic pattern execution\n\n");

  /* open driver */
  sta = rspio4_InitWithOptions(resDesc, VI_TRUE, VI_TRUE, "Simulate=0", &vi);
  /* check return value */
  chk ("rspio4_InitWithOptions");

  if ( VI_SUCCESS == sta )
  {
    runTest();

    /* close driver */
    sta = rspio4_close(vi);
    chk ("rspio4_close");
  }

  printf("\nPress 'Enter' to terminate\n");
```

```
  getchar();

  return 0;
}
```

7.3.2.2 Error Handling

This function checks the return values of a driver call. An error message is issued if there is an error.

```
/* FUNCTION ******************************************************************/
/* checks the return status of a driver call
*******************************************************************************/
static void chk ( char * funcName )
{
  if ( sta != VI_SUCCESS )
  {
    char errorMessage[256];

    rspio4_error_message(vi, sta, errorMessage);
    printf ("%s returned 0x%08X; %s\n", funcName, sta, errorMessage);
  }
}
```

7.3.2.3 Execution of Digital Test

```
/* FUNCTION ******************************************************************/
/* configures the digital test, loads the test and executes it.
*******************************************************************************/
static void runTest ( void )
{
  ViInt32 patternSetResult;

  /* configure stimulus and response levels */
  sta = rspio4_ConfigurePortVoltageRange(vi,
    RSPIO4_MASK_PORT_ALL, RSPIO4_PORT_VOLTAGE_RANGE_2);
  chk ("rspio4_ConfigurePortVoltageRange");

  sta = rspio4_ConfigureStimPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_STIM_MODE_TTL, 3.3, 0.0, 0.1);
  chk ("rspio4_ConfigureStimPort");

  sta = rspio4_ConfigureRespPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_RESP_MODE_HYST, 2.0, 0.8);
  chk ("rspio4_ConfigureRespPort");

  /* configure module for dynamic test and result collection */
  sta = rspio4_ConfigureMode(vi, RSPIO4_VAL_EXECUTE_DYNAMIC,
    RSPIO4_VAL_COLLECT_ALL);
```

```
  chk ("rspio4_ConfigureMode");

  /* create and load the pattern set */
  createPatternSet();

  /* configure the timing */
  sta = rspio4_ConfigurePatternSetTiming(vi, patternSetName, 1.0e-6, 0.5e-6);
  chk ("rspio4_ConfigurePatternSetTiming");

  /* run the test */
  sta = rspio4_ExecutePatternSet(vi, patternSetName, 1000);
  chk ("rspio4_ExecutePatternSet");

  /* fetch pass/fail result */
  sta = rspio4_FetchPatternSetResult(vi, patternSetName, &patternSetResult);
  chk ("rspio4_FetchPatternSetResult");

  if ( RSPIO4_VAL_RESULT_FAIL == patternSetResult )
  {
    printf("Pattern set failed\n");
    diagnosis ();
  }
  else
  {
    printf("Pattern set passed\n");
  }

  /* clear pattern set */
  sta = rspio4_ClearPatternSet(vi, patternSetName);
  chk ("rspio4_ClearPatternSet");
}
```

7.3.2.4 Generation of Pattern Set

```
/* FUNCTION ***************************************************************/
/* creates and loads a pattern set
   channel assignment:

    E_COM          = out1
    E_EC1          = out2
    E_EC2          = out3
    E_EC3          = out4
    N14CR0805170   = out5
    bus            = out20 - out27

    N13HCPL31500   = in1
    N13HCPM31500   = in2
    N13HCPN31500   = in3
    N13HCPO31500   = in4
```

```
*****************************************************************************/
static void createPatternSet ( void )
{
  ViInt32 ph;

  /* create a pattern set */
  sta = rspio4_CreatePatternSet(vi, patternSetName);
  chk ("rspio4_CreatePatternSet");

  /* create a pattern */
  sta = rspio4_CreatePattern(vi, &ph);
  chk ("rspio4_CreatePattern");

  /* start loading */
  sta = rspio4_BeginPatternSetLoading(vi, patternSetName);
  chk ("rspio4_BeginPatternSetLoading");

  /*** 1. pattern : stim all zero, resp don't care */
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IL);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IL);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IL);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IL);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IL);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x0);
  chk ("rspio4_ConfigureGroupOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, i1, RSPIO4_VAL_OPCODE_IOX);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, i2, RSPIO4_VAL_OPCODE_IOX);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, i3, RSPIO4_VAL_OPCODE_IOX);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, i4, RSPIO4_VAL_OPCODE_IOX);
  chk ("rspio4_ConfigureChannelOpcode");
  /* load pattern */
  sta = rspio4_LoadDynamicPattern(vi, patternSetName, ph);
  chk ("rspio4_LoadDynamicPattern");
  // NOTE : previously assigned channels keep their channel opcode in
  //        the following pattern. Therefore only the changes have to be
  //        programmed

  /*** 2. pattern: N14CR0805170 = 1, bus = 01 */
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IH);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x01);
  chk ("rspio4_ConfigureGroupOpcode");
```

```
/* load pattern */
sta = rspio4_LoadDynamicPattern(vi, patternSetName, ph);
chk ("rspio4_LoadDynamicPattern");

/*** 3. pattern: all stim tristate, bus = 02 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x02);
chk ("rspio4_ConfigureGroupOpcode");
/* load pattern */
sta = rspio4_LoadDynamicPattern(vi, patternSetName, ph);
chk ("rspio4_LoadDynamicPattern");

/*** 4. pattern: all stim = 0, bus = 03 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x03);
chk ("rspio4_ConfigureGroupOpcode");
/* load pattern */
sta = rspio4_LoadDynamicPattern(vi, patternSetName, ph);
chk ("rspio4_LoadDynamicPattern");

/*** 5. pattern: N14CR0805170 = 1, bus = 04 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IH);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x04);
chk ("rspio4_ConfigureGroupOpcode");
/* load pattern */
sta = rspio4_LoadDynamicPattern(vi, patternSetName, ph);
chk ("rspio4_LoadDynamicPattern");

/*** 6. pattern: all stim tristate, bus = 05 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
```

```
sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x05);
chk ("rspio4_ConfigureGroupOpcode");
/* load pattern */
sta = rspio4_LoadDynamicPattern(vi, patternSetName, ph);
chk ("rspio4_LoadDynamicPattern");
/*** 7. pattern: all stim = 0, bus = 06, resp N13HCPN31500 = 1 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x06);
chk ("rspio4_ConfigureGroupOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, i3, RSPIO4_VAL_OPCODE_OH);
chk ("rspio4_ConfigureChannelOpcode");
/* load pattern */
sta = rspio4_LoadDynamicPattern(vi, patternSetName, ph);
chk ("rspio4_LoadDynamicPattern");

/* ... etc ... */

/* last pattern : all tristate / don't care */
sta = rspio4_ConfigureGroupOpcode(vi, ph, out, RSPIO4_VAL_GROUP_IGNORE, 0x0);
chk ("rspio4_ConfigureGroupOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, in, RSPIO4_VAL_GROUP_IGNORE, 0x0);
chk ("rspio4_ConfigureGroupOpcode");
/* load pattern */
sta = rspio4_LoadDynamicPattern(vi, patternSetName, ph);
chk ("rspio4_LoadDynamicPattern");

/* loading finished */
sta = rspio4_EndPatternSetLoading(vi, patternSetName);
chk ("rspio4_EndPatternSetLoading");

/* pattern is no longer used */
sta = rspio4_ClearPattern(vi, ph);
chk ("rspio4_ClearPattern");
```

```
}
```

7.3.2.5 Evaluation of Failed Patterns

```
/* FUNCTION ***************************************************************/
/* reads information about pattern set failures and prints it to stdout
*****************************************************************************/
static void diagnosis ( void )
{
  ViInt32 executedPatternCount;
  ViInt32 numChannels = 4;  /* in1 - in4 */
  ViInt32 * pResults;
  ViInt32 * pData;
  ViInt32 * pPatterns;
  ViInt32 actualSize;
  int pattern;
  int channel;
  int idx;
  int failedPatterns;

  sta = rspio4_GetPatternSetExecutedPatternCount (vi, patternSetName,
    &executedPatternCount);
  chk ("rspio4_GetPatternSetExecutedPatternCount");

  printf("%d patterns executed\n", executedPatternCount);

  pResults  = calloc(numChannels * executedPatternCount, sizeof(ViInt32));
  pData     = calloc(numChannels * executedPatternCount, sizeof(ViInt32));
  pPatterns = calloc(executedPatternCount, sizeof(ViInt32));

  /* upload pattern results */
  sta = rspio4_FetchDynamicPatternListResults (vi, patternSetName,
    0, executedPatternCount, executedPatternCount, pPatterns, &actualSize);
  chk ("rspio4_FetchDynamicPatternListResults");

  /* calculate number of failed patterns */
  failedPatterns = 0;
  for ( pattern = 0; pattern < executedPatternCount; pattern++ )
  {
    if ( RSPIO4_VAL_RESULT_FAIL == pPatterns[pattern] )
    {
      failedPatterns++;
    }
  }

  printf("%d patterns failed\n", failedPatterns);

  /* upload data and results for in1 - in4 */
```

```
sta = rspio4_FetchDynamicChannelListPatternData(vi, patternSetName,
  0, executedPatternCount, "in1,in2,in3,in4",
  numChannels * executedPatternCount, pData, &actualSize);
chk ("rspio4_FetchDynamicChannelListPatternData");

sta = rspio4_FetchDynamicChannelListPatternResults (vi, patternSetName,
  0, executedPatternCount, "in1,in2,in3,in4",
  numChannels * executedPatternCount, pResults, &actualSize);
chk ("rspio4_FetchDynamicChannelListPatternResults");
for ( channel = 0; channel < numChannels; channel ++ )
{
  printf("\nin%d :\n", channel+1);

  /* data */
  printf("- Data    : ");
  for ( pattern = 0; pattern < executedPatternCount; pattern++ )
  {
    idx = pattern * numChannels + channel;
    switch ( pData[idx] )
    {
      case RSPIO4_VAL_DATA_HIGH:
        printf("1");
        break;
      case RSPIO4_VAL_DATA_LOW:
        printf("0");
        break;
      default:
        printf("?");
        break;
    }
  }
  printf("\n");

  /* results */
  printf("- Results : ");
  for ( pattern = 0; pattern < executedPatternCount; pattern++ )
  {
    idx = pattern * numChannels + channel;
    switch ( pResults[idx] )
    {
      case RSPIO4_VAL_RESULT_PASS:
        printf(" ");
        break;
      case RSPIO4_VAL_RESULT_FAIL:
        printf("X");
        break;
      default:
        printf("?");
        break;
    }
```

```
    }
    printf("\n");
  }

  free(pResults);
  free(pData);
  free(pPatterns);
}
```

7.3.3 Static Pattern Execution with IVI Digital

7.3.3.1 Main Function

```
/* Example using IVI functions for static pattern execution */
#include <ansi_c.h>
#include "rspio4.h"

/* adapt the resource descriptor to your test system! */
static char resDesc[] = "PXI5::11::INSTR";

/* channel names */
static char o1[] = "OUT1";
static char o2[] = "OUT2";
static char o3[] = "OUT3";
static char o4[] = "OUT4";
static char o5[] = "OUT5";

static char bus[] = "OUT20-OUT27";
static char out[] = "OUT1-OUT31";
static char i1[] = "IN1";
static char i2[] = "IN2";
static char i3[] = "IN3";
static char i4[] = "IN4";

static char in[] = "IN1-IN32";

static ViStatus sta;
static ViSession vi;

/* prototypes */
static void chk ( char * funcName );
static void runTest ( void );
static void executePatternSet ( void );
static void executePattern ( ViInt32 patternHandle, ViInt32 patternIdx );
static void diagnosis ( void );

/* FUNCTION ****************************************************************/
```

```
/* loads the driver and runs the test
*****************************************************************************/
int main (int argc, char *argv[])
{
  printf("Example using IVI functions for static pattern execution\n\n");

  /* open driver */
  sta = rspio4_InitWithOptions(resDesc, VI_TRUE, VI_TRUE, "Simulate=0", &vi);
  /* check return value */
  chk ("rspio4_InitWithOptions");

  if ( VI_SUCCESS == sta )
  {
    runTest();

    /* close driver */
    sta = rspio4_close(vi);
    chk ("rspio4_close");
  }

  printf("\nPress 'Enter' to terminate\n");
  getchar();

  return 0;
}
```

7.3.3.2 Error Handling

This function checks the return values of a driver call. An error message is issued if there is an error.

```
/* FUNCTION ******************************************************************/
/* checks the return status of a driver call
*****************************************************************************/
static void chk ( char * funcName )
{
  if ( sta != VI_SUCCESS )
  {
    char errorMessage[256];

    rspio4_error_message(vi, sta, errorMessage);
    printf ("%s returned 0x%08X; %s\n", funcName, sta, errorMessage);
  }
}
```

7.3.3.3 Execution of Digital Test

```
/* FUNCTION ******************************************************************/
/* configures the digital test, loads the test and executes it.
```

```
*****************************************************************************/
static void runTest ( void )
{
  /* configure stimulus and response levels */
  sta = rspio4_ConfigurePortVoltageRange(vi,
    RSPIO4_MASK_PORT_ALL, RSPIO4_PORT_VOLTAGE_RANGE_2);
  chk ("rspio4_ConfigurePortVoltageRange");

  sta = rspio4_ConfigureStimPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_STIM_MODE_TTL, 3.3, 0.0, 0.1);
  chk ("rspio4_ConfigureStimPort");

  sta = rspio4_ConfigureRespPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_RESP_MODE_HYST, 2.0, 0.8);
  chk ("rspio4_ConfigureRespPort");

  /* configure module for static test and result collection */
  sta = rspio4_ConfigureMode(vi, RSPIO4_VAL_EXECUTE_STATIC,
    RSPIO4_VAL_COLLECT_ALL);
  chk ("rspio4_ConfigureMode");

  /* configure the timing */
  sta = rspio4_ConfigureStaticResponseDelay (vi, 0.5e-6);
  chk ("rspio4_ConfigureStaticResponseDelay");

  /* execute the pattern set */
  executePatternSet();
}
```

7.3.3.4 Execution of a Pattern Set

```
/* FUNCTION ******************************************************************/
/* creates and executes a pattern set
   channel assignment:

    E_COM          = out1
    E_EC1          = out2
    E_EC2          = out3
    E_EC3          = out4
    N14CR0805170   = out5
    bus            = out20 - out27

    N13HCPL31500   = in1
    N13HCPM31500   = in2
    N13HCPN31500   = in3
    N13HCPO31500   = in4
*****************************************************************************/
static void executePatternSet ( void )
{
```

```
ViInt32 ph;
ViInt32 patternIdx = 1;

/* create a pattern */
sta = rspio4_CreatePattern(vi, &ph);
chk ("rspio4_CreatePattern");

/*** 1. pattern : stim all zero, resp don't care */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x0);
chk ("rspio4_ConfigureGroupOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, i1, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, i2, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, i3, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, i4, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");

executePattern(ph, patternIdx++);

// NOTE : previously assigned channels keep their channel opcode in
//        the following pattern. Therefore only the changes have to be
//        programmed

/*** 2. pattern: N14CR0805170 = 1, bus = 01 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IH);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x01);
chk ("rspio4_ConfigureGroupOpcode");

executePattern(ph, patternIdx++);
/*** 3. pattern: all stim tristate, bus = 02 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IOX);
```

```
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x02);
chk ("rspio4_ConfigureGroupOpcode");

executePattern(ph, patternIdx++);

/*** 4. pattern: all stim = 0, bus = 03 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IL);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x03);
chk ("rspio4_ConfigureGroupOpcode");

executePattern(ph, patternIdx++);

/*** 5. pattern: N14CR0805170 = 1, bus = 04 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IH);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x04);
chk ("rspio4_ConfigureGroupOpcode");

executePattern(ph, patternIdx++);

/*** 6. pattern: all stim tristate, bus = 05 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IOX);
chk ("rspio4_ConfigureChannelOpcode");
sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x05);
chk ("rspio4_ConfigureGroupOpcode");

executePattern(ph, patternIdx++);

/*** 7. pattern: all stim = 0, bus = 06, resp N13HCPN31500 = 1 */
sta = rspio4_ConfigureChannelOpcode(vi, ph, o1, RSPIO4_VAL_OPCODE_IL);
```

```
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o2, RSPIO4_VAL_OPCODE_IL);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o3, RSPIO4_VAL_OPCODE_IL);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o4, RSPIO4_VAL_OPCODE_IL);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, o5, RSPIO4_VAL_OPCODE_IL);
  chk ("rspio4_ConfigureChannelOpcode");
  sta = rspio4_ConfigureGroupOpcode(vi, ph, bus, RSPIO4_VAL_GROUP_INPUT, 0x06);
  chk ("rspio4_ConfigureGroupOpcode");
  sta = rspio4_ConfigureChannelOpcode(vi, ph, i3, RSPIO4_VAL_OPCODE_OH);
  chk ("rspio4_ConfigureChannelOpcode");

  executePattern(ph, patternIdx++);

  /* ... etc ... */

  /* last pattern : all tristate / don't care */
  sta = rspio4_ConfigureGroupOpcode(vi, ph, out, RSPIO4_VAL_GROUP_IGNORE, 0x0);
  chk ("rspio4_ConfigureGroupOpcode");
  sta = rspio4_ConfigureGroupOpcode(vi, ph, in, RSPIO4_VAL_GROUP_IGNORE, 0x0);
  chk ("rspio4_ConfigureGroupOpcode");

  executePattern(ph, patternIdx++);

  /* pattern is no longer used */
  sta = rspio4_ClearPattern(vi, ph);
  chk ("rspio4_ClearPattern");
}
```

7.3.3.5 Execution of a Single Pattern

```
/* FUNCTION ******************************************************************/
/* executes a single pattern
*******************************************************************************/
static void executePattern ( ViInt32 patternHandle, ViInt32 patternIdx )
{
  ViInt32 patternResult;

  sta = rspio4_ExecuteStaticPattern(vi, patternHandle);
  chk ("rspio4_ExecuteStaticPattern");

  sta = rspio4_FetchStaticPatternResult (vi, &patternResult);
  chk ("rspio4_FetchStaticPatternResult");

  if ( RSPIO4_VAL_RESULT_FAIL == patternResult )
  {
    printf("Pattern %d failed\n", patternIdx);
```

```
    diagnosis ();
  }
  else
  {
    printf("Pattern %d passed\n", patternIdx);
  }
}
```

7.3.3.6 Evaluation of a Failed Pattern

```
/* FUNCTION ******************************************************************/
/* reads information about pattern failures and prints it to stdout
*******************************************************************************/
static void diagnosis ( void )
{
  ViInt32 numChannels = 4;  /* in1 - in4 */
  ViInt32 results[4];
  ViInt32 data[4];
  ViInt32 actualSize;
  int channel;

  /* upload data and results for in1 - in4 */
  sta = rspio4_FetchStaticChannelListData(vi, "in1,in2,in3,in4",
    numChannels, data, &actualSize);
  chk ("rspio4_FetchStaticChannelListData");

  sta = rspio4_FetchStaticChannelListResults(vi, "in1,in2,in3,in4",
    numChannels, results, &actualSize);
  chk ("rspio4_FetchStaticChannelListResults");

  for ( channel = 0; channel < numChannels; channel ++ )
  {
    printf("  in%d ", channel+1);

    switch ( data[channel] )
    {
      case RSPIO4_VAL_DATA_HIGH:
        printf("high ");
        break;
      case RSPIO4_VAL_DATA_LOW:
        printf("low  ");
        break;
      default:
        printf("?    ");
        break;
    }

    switch ( results[channel] )
    {
```

```
      case RSPIO4_VAL_RESULT_PASS:
        printf("pass");
        break;
      case RSPIO4_VAL_RESULT_FAIL:
        printf("fail");
        break;
      default:
        printf("not available");
        break;
    }

    printf("\n");
  }
}
```

7.3.4 Static Pattern Output with Low-Level Driver Functions

7.3.4.1 Main Function

```
/* Example using low level driver functions for static pattern execution */
#include <utility.h>
#include <ansi_c.h>

#include "rspio4.h"

#define PATTERN_COUNT       8

/* adapt the resource descriptor to your test system! */
static char resDesc[] = "PXI5::11::INSTR";

static ViUInt32 stimData[PATTERN_COUNT] = {
  0x00000000,
  0x00000010,
  0x0000001F, /* tri state */
  0x00000000,
  0x00000010,
  0x0000001F, /* tri state */
  0x00000000,
  /* etc. */
  0xFFFFFFFF
};

static ViUInt16 portEnable[PATTERN_COUNT] = {
  0xFF,
  0xFF,
  0xFC, /* port 0 and port 1 tri state (OUT1 to OUT8) */
  0xFF,
```

```
  0xFF,
  0xFC, /* port 0 and port 1 tri state (OUT1 to OUT8) */
  0xFF,
  /* etc. */
  0x00  /* port 0 to port 7 tri state (OUT1 to OUT32) */
};

static ViStatus sta;
static ViSession vi;

/* prototypes */
static void chk ( char * funcName );
static void runTest ( void );
static void executePatternSet( void );

/* FUNCTION ******************************************************************/
/* loads the driver and runs the test
*******************************************************************************/
int main(int argc, char *argv[])
{
  printf("Use of low level driver functions for static pattern execution\n\n");

  /* open a session to the device driver */
  sta = rspio4_InitWithOptions(resDesc, VI_TRUE, VI_TRUE, "Simulate=0", & vi);
  /* check return value */
  chk ("rspio4_InitWithOptions");

  if (VI_SUCCESS == sta)
  {
    runTest();

    /* close the driver */
    sta = rspio4_close(vi);
    chk ("rspio4_close");
  }
  printf("\nPress 'Enter' to terminate\n");
  getchar();

  return 0;
}
```

7.3.4.2 Error Handling

This function checks the return values of a driver call. A message is issued if there is an error.

```
/* FUNCTION ******************************************************************/
/* checks the return status of a driver call
*******************************************************************************/
```

```
static void chk ( char * funcName )
{
  if ( sta != VI_SUCCESS )
  {
    char errorMessage[256];

    rspio4_error_message(vi, sta, errorMessage);
    printf ("%s returned 0x%08X; %s\n", funcName, sta, errorMessage);
  }
}
```

7.3.4.3 Execution of Digital Test

```
/* FUNCTION ****************************************************************/
/* configures the digital test and executes it
*******************************************************************************/
static void runTest ( void )
{
  ViReal64 voltageHigh = 3.3;
  ViReal64 voltageLow = 0.0;
  ViReal64 currentLimit = 0.1;
  ViReal64 thresholdHigh = 2.0;
  ViReal64 thresholdLow = 0.8;

  /* set voltage range;   */
  sta = rspio4_ConfigurePortVoltageRange(vi,
    RSPIO4_MASK_PORT_ALL, RSPIO4_PORT_VOLTAGE_RANGE_2);
  chk ("rspio4_ConfigurePortVoltageRange");

  /* configure driver voltage levels */
  sta = rspio4_ConfigureStimPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_STIM_MODE_TTL, voltageHigh, voltageLow, currentLimit);
  chk ("rspio4_ConfigureStimPort");

  /* configure sensor: set all ports, use hysteresis comparator mode */
  sta = rspio4_ConfigureRespPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_RESP_MODE_HYST, thresholdHigh, thresholdLow);
  chk ("rspio4_ConfigureRespPort");

  /* configure module for dynamic test and result collection */
  sta = rspio4_ConfigureMode(vi, RSPIO4_VAL_EXECUTE_STATIC,
    RSPIO4_VAL_COLLECT_ALL);
  chk ("rspio4_ConfigureMode");

  /* configure 32 bit wide dynamic operation: OUT1 .. OUT32 will be operated
     dynamically; no static channels */
  sta = rspio4_ConfigureStimMode (vi, RSPIO4_VAL_CTRL_STATIC);
  chk ("rspio4_ConfigureStimMode");
```

```
  /* configure 32 bit wide dynamic operation */
  sta = rspio4_ConfigureRespMode(vi, RSPIO4_VAL_CTRL_STATIC);
  chk ("rspio4_ConfigureRespMode");

  /* execute the pattern set */
  executePatternSet();
}
```

7.3.4.4 Execution of a Pattern Set

```
/* FUNCTION ******************************************************************/
/* executes the pattern set
*******************************************************************************/
static void executePatternSet( void )
{
  int      loopIdx, portIdx;
  ViUInt32 response;
  ViUInt32 portMask;
  ViChar   enableInfo[9];
  ViUInt32 channelMask = 0xFFFFFFFF;
  ViUInt16 enablePortMask = RSPIO4_MASK_PORT_ALL;
  ViReal64 uutResponseDelay = 0.001;

  printf("Idx | Stimulus   | Enable     | Response  \n");
  printf("----+------------+------------+-----------\n");

  for (loopIdx = 0; loopIdx < PATTERN_COUNT; loopIdx++)
  {
    sta = rspio4_SetDoutPort(vi, channelMask,
       stimData[loopIdx], enablePortMask, portEnable[loopIdx]);
    chk ("rspio4_SetDoutPort");

    Delay(uutResponseDelay);

    sta = rspio4_GetDinState(vi, &response);
    chk ("rspio4_GetDinState");

    portMask = RSPIO4_MASK_PORT0;
    for (portIdx = 7; portIdx >= 0; portIdx--)
    {
      if (portEnable[loopIdx] & portMask)
      {
        enableInfo[portIdx] = 'F';
      }
      else
      {
        enableInfo[portIdx] = '0';
      }
      portMask = portMask << 1;
```

```
    }

    printf("%3d | 0x%08X | 0x%s | 0x%08X\n", loopIdx,
      stimData[loopIdx], enableInfo, response);
  }
}
```

7.3.5 Dynamic Pattern Execution with Low-Level Driver Functions

7.3.5.1 Main Function

```
/* Example using low level driver functions for dynamic pattern execution */
#include <ansi_c.h>
#include "rspio4.h"

#define PATTERN_COUNT       8
#define PATTERN_PERIOD      1.0e-6
#define FETCH_TIMEOUT       0.1

/* adapt the resource descriptor to your test system! */
static char resDesc[] = "PXI5::11::INSTR";

static ViUInt32 stimData[PATTERN_COUNT] = {
  0x00000000,
  0x00000010,
  0x0000001F, /* tri state */
  0x00000000,
  0x00000010,
  0x0000001F, /* tri state */
  0x00000000,
  /* etc. */
  0xFFFFFFFF
};

static ViUInt32 stimTristate[PATTERN_COUNT] = {
  0x00000000,
  0x00000000,
  0x00000003, /* port 0 and port 1 tri state (OUT1 to OUT8) */
  0x00000000,
  0x00000000,
  0x00000003, /* port 0 and port 1 tri state (OUT1 to OUT8) */
  0x00000000,
  /* etc. */
  0x000000FF  /* port 0 to port 7 tri state (OUT1 to OUT32) */
};

static ViStatus sta;
```

```
static ViSession vi;
static ViUInt16 memId;

/* data buffer */
static RSPIO4_DATA_32BIT_TRISTATE stimulus[PATTERN_COUNT];
static RSPIO4_DATA_32BIT          response[PATTERN_COUNT];

/* prototypes */
static void chk ( char * funcName );
static void runTest ( void );
static void createPatternSet ( void );
/* FUNCTION ******************************************************************/
/* loads the driver and runs the test
*******************************************************************************/
int main(int argc, char *argv[])
{
  printf("Use of low level driver functions for dynamic pattern execution\n\n");

  /* open a session to the device driver */
  sta = rspio4_InitWithOptions(resDesc, VI_TRUE, VI_TRUE, "Simulate=0", & vi);
  /* check return value */
  chk ("rspio4_InitWithOptions");

  if (VI_SUCCESS == sta)
  {
    runTest();

    /* close the driver */
    sta = rspio4_close(vi);
    chk ("rspio4_close");
  }

  printf("\nPress 'Enter' to terminate\n");
  getchar();

  return 0;
}
```

7.3.5.2 Error Handling

This function checks the return values of a driver call. A message is issued if there is an error.

```
/* FUNCTION ******************************************************************/
/* checks the return status of a driver call
*******************************************************************************/
static void chk ( char * funcName )
{
  if ( sta != VI_SUCCESS )
  {
```

```
    char errorMessage[256];

    rspio4_error_message(vi, sta, errorMessage);
    printf ("%s returned 0x%08X; %s\n", funcName, sta, errorMessage);
  }
}
```

7.3.5.3 Execution of Digital Test

```
/* FUNCTION ****************************************************************/
/* configures the digital test, loads the test and executes it.
*****************************************************************************/
static void runTest ( void )
{
  ViUInt32 respByteCount;
  ViInt32 loopIdx;

  ViReal64 voltageHigh = 3.3;
  ViReal64 voltageLow = 0.0;
  ViReal64 currentLimit = 0.1;
  ViReal64 thresholdHigh = 2.0;
  ViReal64 thresholdLow = 0.8;

  ViReal64 triggerDelayStim = 0.0;
  ViReal64 triggerDelayResp = PATTERN_PERIOD / 2.0;

  /* set voltage range;   */
  sta = rspio4_ConfigurePortVoltageRange(vi,
    RSPIO4_MASK_PORT_ALL, RSPIO4_PORT_VOLTAGE_RANGE_2);
  chk ("rspio4_ConfigurePortVoltageRange");

  /* configure driver voltage levels */
  sta = rspio4_ConfigureStimPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_STIM_MODE_TTL, voltageHigh, voltageLow, currentLimit);
  chk ("rspio4_ConfigureStimPort");

  /* configure sensor: set all ports, use hysteresis comparator mode */
  sta = rspio4_ConfigureRespPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_RESP_MODE_HYST, thresholdHigh, thresholdLow);
  chk ("rspio4_ConfigureRespPort");

  /* configure module for dynamic test and result collection */
  sta = rspio4_ConfigureMode(vi, RSPIO4_VAL_EXECUTE_DYNAMIC,
    RSPIO4_VAL_COLLECT_ALL);
  chk ("rspio4_ConfigureMode");

  /* configure 32 bit wide dynamic operation: OUT1 to OUT32 will be operated
     dynamically; no static channels */
  sta = rspio4_ConfigureStimMode (vi, RSPIO4_VAL_CTRL_32BIT_TRISTATE);
```

```
  chk ("rspio4_ConfigureStimMode");

  /* configure 32 bit wide dynamic operation */
  sta = rspio4_ConfigureRespMode(vi, RSPIO4_VAL_CTRL_32BIT);
  chk ("rspio4_ConfigureRespMode");

  /* configure stimulus timing */
  sta = rspio4_ConfigureStimTiming(vi, triggerDelayStim,
    PATTERN_PERIOD, PATTERN_COUNT);
  chk ("rspio4_ConfigureStimTiming");

  /* configure response timing */
  sta = rspio4_ConfigureRespTiming(vi, triggerDelayResp,
    PATTERN_PERIOD, PATTERN_COUNT);
  chk ("rspio4_ConfigureRespTiming");

  /* create and load the pattern set */
  createPatternSet();

  /* start pattern execution */
  sta = rspio4_ExecutePattern(vi, memId);
  chk ("rspio4_ExecutePattern");

  /* read the response data */
  sta = rspio4_FetchPatternResponseData(vi, sizeof(response),
    (ViAddr *)response, FETCH_TIMEOUT, & respByteCount);
  chk ("rspio4_FetchPatternResponseData");

  /* evaluate response data */
  printf("Response data:\n");

  for (loopIdx = 0; loopIdx < PATTERN_COUNT; loopIdx++)
  {
    printf("%2d 0x%08X\n", loopIdx, response[loopIdx].data);
  }

  /* free stimulus memory */
  sta = rspio4_DiscardData(vi, memId);
  chk ("rspio4_DiscardData");
}
```

7.3.5.4 Generation of a Pattern Set

```
/* FUNCTION ******************************************************************/
/* creates and loads a pattern set
*******************************************************************************/
static void createPatternSet ( void )
{
  int      loopIdx, portIdx;
```

```
  ViUInt32 portMask;
  ViChar   triStateInfo[9];

  /* generate stimulus data */
  printf("Stimulus data:\n");

  for (loopIdx = 0; loopIdx < PATTERN_COUNT; loopIdx++)
  {
    /* add index information to OUT20 to OUT27 */
    stimulus[loopIdx].data = stimData[loopIdx] | (loopIdx << 19);
    stimulus[loopIdx].tristate = stimTristate[loopIdx];

    portMask = RSPIO4_MASK_PORT0;
    for (portIdx = 7; portIdx >= 0; portIdx--)
    {
      if (stimulus[loopIdx].tristate & portMask)
      {
        triStateInfo[portIdx] = 'F';
      }
      else
      {
        triStateInfo[portIdx] = '0';
      }
      portMask = portMask << 1;
    }

    printf("%2d 0x%08X\n", loopIdx, stimulus[loopIdx].data);
    printf("   0x%s\n", triStateInfo);
  }

  /* load data to stimulus RAM */
  sta = rspio4_LoadData (vi, (ViAddr)s_stimData, sizeof(s_stimData),
    RSPIO4_VAL_DATA_TYPE_STIM, & memId);
  chk ("rspio4_LoadData");
}
```

7.3.6 Triggered Pattern Execution

In this example, a pulse on trigger line PXI0 of the TSVP backplane triggers output of the pattern. The trigger pulse is generated by the R&S TS-PIO4 module itself. Other measurement modules in the TSVP frame can also use this signal to perform a triggered measurement. It is, of course, also possible to start other R&S TS-PIO4 modules in the system using this signal. In the example, the stimulus logic and response logic (IT1 and IT2) are started by the same signal (PXI0). In addition, the output clock of the stimulus logic (internal trigger logic block 1 or IT1) is routed to pin XTO of the front connector.

7.3.6.1 Main Program

```
/* Example using low level driver functions for triggered execution */
#include <ansi_c.h>
#include "rspio4.h"

#define PATTERN_COUNT       8
#define PATTERN_PERIOD      1.0e-6

/* adapt the resource descriptor to your test system! */
static char resDesc[] = "PXI5::11::INSTR";

/* data buffer */
static RSPIO4_DATA_32BIT response[PATTERN_COUNT];
static RSPIO4_DATA_32BIT stimulus[PATTERN_COUNT] = {
  0x00000000,
  0x00000001,
  0x00000002,
  0x00000003,
  0x00000004,
  0x00000005,
  0x00000006,
  0x00000007
};

static ViStatus sta;
static ViSession vi;
static ViUInt16 memId;

/* prototypes */
static void chk ( char * funcName );
static void runTest ( void );

/* FUNCTION ******************************************************************/
/* loads the driver and runs the test
*******************************************************************************/
int main(int argc, char *argv[])
{
  printf("Use of low level driver functions for triggered execution\n\n");

  /* open a session to the device driver */
  sta = rspio4_InitWithOptions(resDesc, VI_TRUE, VI_TRUE, "Simulate=0", & vi);
  /* check return value */
  chk ("rspio4_InitWithOptions");

  if (VI_SUCCESS == sta)
  {
    runTest();

    /* close the driver */
```

```
    sta = rspio4_close(vi);
    chk ("rspio4_close");
  }

  printf("\nPress 'Enter' to terminate\n");
  getchar();

  return 0;
}
```

7.3.6.2 Error Handling

```
/* FUNCTION ****************************************************************/
/* checks the return status of a driver call
*****************************************************************************/
static void chk ( char * funcName )
{
  if ( sta != VI_SUCCESS )
  {
    char errorMessage[256];

    rspio4_error_message(vi, sta, errorMessage);
    printf ("%s returned 0x%08X; %s\n", funcName, sta, errorMessage);
  }
}
```

7.3.6.3 Triggered Digital Test

```
/* FUNCTION ****************************************************************/
/* configures the digital test, loads the test and executes it.
*****************************************************************************/
static void runTest ( void )
{
  ViUInt32 respByteCount;
  ViInt32 loopIdx;

  /* set voltage range;   */
  sta = rspio4_ConfigurePortVoltageRange(vi,
    RSPIO4_MASK_PORT_ALL, RSPIO4_PORT_VOLTAGE_RANGE_2);
  chk ("rspio4_ConfigurePortVoltageRange");

  /* configure driver voltage levels */
  sta = rspio4_ConfigureStimPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_STIM_MODE_TTL, 3.3, 0.0, 0.1);
  chk ("rspio4_ConfigureStimPort");

  /* configure sensor: set all ports, use hysteresis comparator mode */
  sta = rspio4_ConfigureRespPort(vi, RSPIO4_MASK_PORT_ALL,
    RSPIO4_RESP_MODE_HYST, 2.0, 0.8);
```

```
chk ("rspio4_ConfigureRespPort");

/* configure module for dynamic test and result collection */
sta = rspio4_ConfigureMode(vi, RSPIO4_VAL_EXECUTE_DYNAMIC,
  RSPIO4_VAL_COLLECT_ALL);
chk ("rspio4_ConfigureMode");

/* configure 32 bit wide dynamic operation */
sta = rspio4_ConfigureStimMode(vi, RSPIO4_VAL_CTRL_32BIT);
chk ("rspio4_ConfigureStimMode");

sta = rspio4_ConfigureRespMode(vi, RSPIO4_VAL_CTRL_32BIT);
chk ("rspio4_ConfigureRespMode");

/* configure stimulus timing */
sta = rspio4_ConfigureStimTiming(vi, 0.0, PATTERN_PERIOD, PATTERN_COUNT);
chk ("rspio4_ConfigureStimTiming");

/* configure response timing */
sta = rspio4_ConfigureRespTiming(vi, PATTERN_PERIOD / 2.0,
  PATTERN_PERIOD, PATTERN_COUNT);
chk ("rspio4_ConfigureRespTiming");

/* load data to stimulus RAM */
sta = rspio4_LoadData (vi, (ViAddr)stimulus, sizeof(stimulus),
  RSPIO4_VAL_DATA_TYPE_STIM, & memId);
chk ("rspio4_LoadData");

/* connect all outputs to inputs */
sta = rspio4_ConnectInOut (vi, RSPIO4_MASK_PORT_ALL, RSPIO4_MASK_PORT_ALL);
chk ("rspio4_ConnectInOut");

/* configure trigger logic blocks to output the clock pulses  */
sta = rspio4_ConfigItTrigOut(vi, RSPIO4_IT_MASK_IT1 | RSPIO4_IT_MASK_IT2,
  RSPIO4_VAL_TRIG_IT_OUT);
chk ("rspio4_ConfigItTrigOut");

/* configure XTO to output the signal of trigger logic block 1 (stimulus) */
sta = rspio4_ConfigXTO(vi, RSPIO4_VAL_TRIG_IT1);
chk ("rspio4_ConfigXTO");


/* general purpose trigger should output a pulse at PXI0 */
sta = rspio4_ConfigPxiTrigOut(vi, RSPIO4_TRIG_MASK_PXI0, RSPIO4_VAL_TRIG_GP,
  RSPIO4_TRIG_MASK_PXI0, RSPIO4_TRIG_MASK_PXI0);
chk ("rspio4_ConfigPxiTrigOut");


/* stimmulus and response should be triggert by PXI0 */
sta = rspio4_ConfigHWTriggerInput(vi, RSPIO4_IT_MASK_IT1 | RSPIO4_IT_MASK_IT2,
```

```
    RSPIO4_TRIG_MASK_PXI0, RSPIO4_TRIG_MASK_PXI0,
                                        RSPIO4_VAL_FALLING_EDGE);
  chk ("rspio4_ConfigHWTriggerInput");

  /* load the stimulus buffer for triggered pattern generation */
  sta = rspio4_LoadStimBuffer (vi, memId);
  chk ("rspio4_LoadStimBuffer");

  /* arm trigger logic blocks */
  sta = rspio4_EnableHWTrigger(vi, RSPIO4_IT_MASK_IT1 | RSPIO4_IT_MASK_IT2,
    RSPIO4_IT_MASK_IT1 | RSPIO4_IT_MASK_IT2);
  chk ("rspio4_EnableHWTrigger");

  /* wait until all pending hardware configurations have settled */
  sta = rspio4_WaitForDebounce(vi, 100);
  chk ("rspio4_WaitForDebounce");

  /* generate the general purpose trigger pulse at PXI0 */
  sta = rspio4_InitiateSWTrigger(vi, RSPIO4_IT_MASK_GP);
  chk ("rspio4_InitiateSWTrigger");

  /* wait until the pattern execution has finished; read the response data */
  sta = rspio4_FetchPatternResponseData(vi, sizeof(response),
    (ViAddr *)response, 0.1, & respByteCount);
  chk ("rspio4_FetchPatternResponseData");

  /* evaluate response data */
  printf("Idx | Stimulus   | Response  \n");
  printf("----------------------------\n");

  for (loopIdx = 0; loopIdx < PATTERN_COUNT; loopIdx++)
  {
    printf("% 3d | 0x%08X | 0x%08X\n",
      loopIdx, stimulus[loopIdx].data, response[loopIdx].data);
  }

  /* free stimulus memory */
  sta = rspio4_DiscardData(vi, memId);
  chk ("rspio4_DiscardData");
}
```

8 Self-Test

The R&S TS-PIO4 digital functional test module has built-in self-test capability. The following tests are implemented:

- LED test
- Power-on test
- TSVP self-test

8.1 LED Test

When the device is switched on, all three LEDs light up for approx. one second. This indicates that the 5 V supply voltage is present and all LEDs are OK. The following statements can be made regarding the different LED states:

LED	Description
One LED does not light up	Hardware problem on the module; LED defective
No LEDs light up	No +5 V supply voltage

8.2 Power-On Test

The power-on test runs at the same time as the LED test. The following statements can be made regarding the different display states of the LEDs:

LED	Description
PWR LED (green) on	All supply voltages are present
PWR LED (green) off	At least one supply voltage is missing
ERR LED (red) off	No errors have been detected
ERR LED (red) on	FPGA loading has failed

8.3 TSVP Self-Test

The TSVP self-test runs an in-depth test on the R&S TS-PIO4 module and generates a detailed log. This is done using the "Self-Test Support Library".

The R&S TS-PSAM analog stimulus and measurement module is used as a measurement unit in the TSVP self-test in order to check trigger line PXI_TRIG[0-7].

Information on starting the self-test and on the sequence of the necessary steps as well as a detailed description of the checked parameters and operations can be found in the *R&S CompactTSVP / R&S PowerTSVP Service Manual*.

9 Interface Description

9.1 R&S TS-PIO4

9.1.1 Connector X10

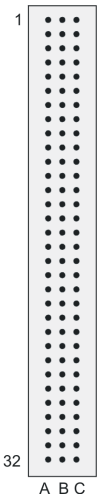


Figure 9-1: Connector X10 (mating side)

Table 9-1: Pin assignment of R&S TS-PIO4 connector X10

	A	B	C
1		AUX1	EXT_CLK
2		AUX2	
3		AUX3	
4		AUX4	
5	OUT1	OUT2	OUT3
6	IN1	IN2	IN3
7	OUT4	OUT5	OUT6
8	IN4	IN5	IN6
9	OUT7	OUT8	GNDNO
10	IN7	IN8	GND
11	OUT9	OUT10	OUT11
12	IN9	IN10	IN11
13	OUT12	OUT13	OUT14

	A	B	C
14	IN12	IN13	IN14
15	OUT15	OUT16	GNDNO
16	IN15	IN16	GND
17	OUT17	OUT18	OUT19
18	IN17	IN18	IN19
19	OUT20	OUT21	OUT22
20	IN20	IN21	IN22
21	OUT23	OUT24	GNDNO
22	IN23	IN24	GND
23	OUT25	OUT26	OUT27
24	IN25	IN26	IN27
25	OUT28	OUT29	OUT30
26	IN28	IN29	IN30
27	OUT31	OUT32	GNDNO
28	IN31	IN32	GND
29	XTO		
30	XTI		
31			GND
32			CHA-GND

9.1.2 Connector X20

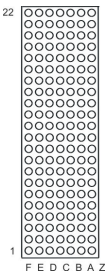


Figure 9-2: Connector X20 (mating side)

Pin	F	E	D	C	B	A	Z	
22		GA0	GA1	GA2	GA3	GA4		
21		OD8						
20		+5V	GND	+5V	AUX1A_COM	AUX2A_COM		
19		AUX1A_COM	AUX2A_COM	+5V	GND	-12V		
18		PXI_TRIG6	GND	PXI_TRIG5	PXI_TRIG4	PXI_TRIG3		J20
17		PXI_CLK10	PO_2	PO_1	GND	PXI_TRIG2		
16		PXI_TRIG7	GND	PO_3	PXI_TRIG0	PXI_TRIG1		
15			+5V	PO_4	GND			
14	NC	AUX1A_NC	AUX1A_NO		AUX3A_NO	AUX3A_NC	NC	C
13	NC	AUX1A_NC	AUX1A_NO		AUX3A_NO	AUX3A_NC	NC	O
12	NP	AUX1A_COM	AUX2A_NO		AUX4A_NO	AUX3A_COM	NP	N
11	NP	AUX1A_COM	AUX2A_NO	IL1	AUX4A_NO	AUX3A_COM	NP	N
10	NC	AUX2A_COM	AUX2A_NC		AUX4A_NC	AUX4A_COM	NC	E
9	NC	AUX2A_COM	AUX2A_NC		AUX4A_NC	AUX4A_COM	NC	C
8	NC	AUX1B_COM	AUX1B_NO		AUX3B_NC	AUX3B_COM	NC	T
7	NC	AUX1B_COM	AUX1B_NC	IL2	AUX3B_NO	AUX3B_COM	NC	O
6	NC	AUX2B_COM	AUX2B_NO		AUX4B_NC	AUX4B_COM	NC	R
5	NC	AUX2B_COM	AUX2B_NC		AUX4B_NO	AUX4B_COM	NC	
4	NC						NC	
3		RSA0	RRST#		GND	RSDO		
2		+12V	RSDI	RSA1		RSCLK		
1		+5V			GND	RCS#		

Figure 9-3: Pin assignment of connector X20

9.1.3 Connector X30

Table 9-2: Assignment of R&S TS-PIO4 connector X30

	E	E	C	B	A
7					
6			GND		
5					
4					
3					
2					
1					

9.1.4 Connector X1 (cPCI Bus)

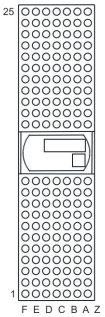


Figure 9-4: Connector X1 (mating side)

Pin	F	E	D	C	B	A	Z	
25	GND	5V	3.3V	ENUM#	REQ64#	5V	GND	
24	GND	ACK64#	AD[0]	V(I/O)	5V	AD[1]	GND	
23	GND	AD[2]	5V	AD[3]	AD[4]	3.3V	GND	
22	GND	AD[5]	AD[6]	3.3V	GND	AD[7]	GND	
21	GND	C/BE[0]#	M66EN	AD[8]	AD[9]	3.3V	GND	
20	GND	AD[10]	AD[11]	V(I/O)	GND	AD[12]	GND	X1
19	GND	AD[13]	GND	AD[14]	AD[15]	3.3V	GND	
18	GND	C/BE[1]#	PAR	3.3V	GND	SERR#	GND	
17	GND	PERR#	GND	IPMB_SDA	IPMB_SCL	3.3V	GND	C
16	GND	LOCK#	STOP#	V(I/O)	GND	DEVSEL#	GND	O
15	GND	TRDY#	BD_SEL#	IRDY#	FRAME#	3.3V	GND	N
12..14				Key Area				N
11	GND	C/BE[2]#	GND	AD[16]	AD[17]	AD[18]	GND	E
10	GND	AD[19]	AD[20]	3.3V	GND	AD[21]	GND	C
9	GND	AD[22]	GND	AD[23]	IDSEL	C/BE[3]#	GND	T
8	GND	AD[24]	AD[25]	V(I/O)	GND	AD[26]	GND	O
7	GND	AD[27]	GND	AD[28]	AD[29]	AD[30]	GND	R
6	GND	AD[31]	CLK	3.3V	GND	REQ#	GND	
5	GND	GNT#	GND	RST#	BSRSV	BSRSV	GND	
4	GND	INTS	INTP	V(I/O)	HEALTHY#	IPMB_PWR	GND	
3	GND	INTD#	5V	INTC#	INTB#	INTA#	GND	
2	GND	TDI	TDO	TMS	5V	TCK	GND	
1	GND	5V	+12V	TRST#	-12V	5V	GND	

Figure 9-5: Pin assignment of connector X1

10 Specifications

The specifications for the R&S TS-PIO4 module are given in the corresponding data sheets.

If discrepancies exist between the data given in this manual and the values in the data sheet, the values in the data sheet take precedence.