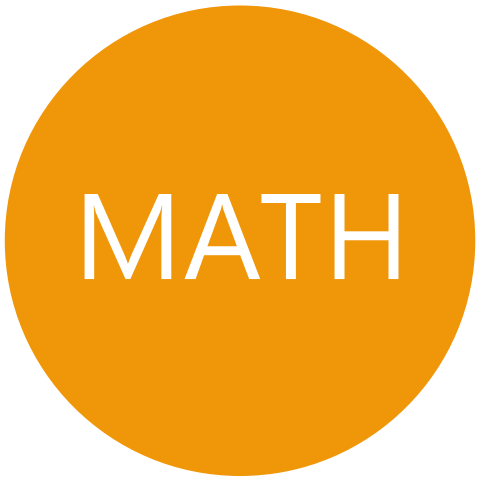
magnis technology plus



MÓDULO HARDWARE (HAL)
Arquitectura y especificaciones

INDICE

1 CONTENIDO

2 INTRODUCCIÓN

Ésta documentación tiene como objetivo brindar al lector una guía general correspondiente al módulo MATH de la aplicación WINFAS 2, y así funcionar de complemento para toda la documentación detallada en el propio código fuente de la aplicación (Diagrama en Bloque, archivo de proyecto, etc).

En ésta documentación no se explicarán detalles correspondientes sobre ingeniería electrónica, adquisición de datos por hardware, lenguaje de programación LabVIEW, herramientas o patrones de diseños brindados por National Instruments, ya que se presupone que una persona con los conocimientos pertinentes de la herramienta y de la disciplina, ya conoce dichos contenidos.

Lo que sí se hará incapié, es en diagramar claramente los componentes y arquitectura que conforman el módulo para facilitar (junto con la documentación embebida en el código fuente) el mantenimiento y escalabilidad del mismo.

3 ARQUITECTURA QMH (QUEUED MESSAGE HANDLER)

3.1 DESCRIPCIÓN

El Módulo Matemático de WINFAS 2 al igual que el módulo de Hardware está desarrollado en base a un patrón de diseño standard de National Instruments llamado Queued Message Hanlder (QMH). Este módulo hace hincapié en el rendimiento de cada uno de sus sub-bloques con la idea de poder mejorar el tiempo de ejecución de cada uno.

3.2 MHL MATEMÁTICO

3.2.1 Descripción

El MATH Loop es un MHL diseñado pura y exclusivamente para chequear, corregir, editar y calcular las operaciones matemáticas necesarias que sean solicitadas por el WINFAS 2. En el siguiente apartado se pretende poner en evidencia como está compuesto internamente este módulo, desde las tareas invocadas por los mensajes hasta el flujo de datos que maneja.

3.2.2 Tareas

3.2.2.1 LOOP

- **Default**: Proceso vacío en el cual el loop puede entrar por un estado erróneo y no producir inconvenientes en el estado del programa.
- **EXIT**: Cierra el loop y termina el VI.

3.2.2.2 ACTION

- **Solve:** Mapea la entrada de valores correspondientes de la Queues con los valores seteados por el usuario. Calcula las medidas auxiliares en primera instancia para luego pasar a calcular el valor final de Resultado con las mediciones solicitadas.

3.2.2.3 UTILITY

- **Config:** Invoca el panel de configuración propio del dispositivo. Este panel tiene la capacidad de poder cambiar, eliminar y agregar nuevos valores ya sea de Constantes, Dimensiones de Patrón, Medidas Auxiliares y Mediciones.
- **Set Config:** Carga la configuración desde afuera.
- **Get Config:** Envía la configuración hacia afuera.

3.2.3 Estructura de Datos

3.2.3.1 ***Variable Data***

Compuesto por un cluster de Transductores, Constantes, Dimensiones de Patrón y Medidas Auxiliares. Es importante destacar que estos valores están almacenados en la memoria del programa y no se guardan en archivos externos con la finalidad de incrementar la performance del sistema.

- Transductores: Array de 2 dimensiones con la información mapeada de los transductores que se quieran utilizar.
- Constantes: Array de 1 dimensión que contiene todas las constantes que el usuario carga, edita o agrega.
- Dimensiones de Patrón: Array de 1 dimensión que contiene todas las constantes que el usuario carga, edita o agrega.
- Medidas Auxiliares: Array de 2 dimensiones generado a partir de las funciones matemáticas que se le han asignado, el cual es posible cargarle, editarle o agregarle auxiliares.

3.2.3.2 ***Array Parsed Expression***

Compuesto por un Array de cluster con toda la información necesaria para poder realizar las operaciones de las Medidas Auxiliares, las Mediciones y los Resultados

- RPN: Notación Reversa Polaca. Este método algebraico alternativo de introducción de datos es indispensable para el cálculo de las funciones matemáticas. En cada cluster se encuentran armado cada RPN correspondiente a la función que aloje el mismo.
- Constantes: Array de constantes que son extraídas de la función del cluster.
- Variables: Array de variables que son extraídas de la función del cluster.
- Formula: Función original a la cual se le pretende utilizar para cálculos de mediciones o auxiliares.
- Tipo de Medición: Tipo de resultado que se le aplican a las Mediciones. Este valor es válido solo para clusters de Mediciones, es decir no tiene efecto en cluster de auxiliares.
- Med?: Bandera que sirve de referencia para poder diferenciar entre cluster de auxiliares y clusters de mediciones.
- Descripción: Descripción de las Mediciones que se realizan. Este valor es válido solo para clusters de Mediciones.
- Nominal: Valor nominal que se pretende conseguir del resultado de la medición. Este valor es válido solo para clusters de Mediciones.
- Tol Sup: Tolerancia superior del valor nominal. Este valor es válido solo para clusters de Mediciones.
- Pre TS: Tolerancia pre-superior del valor nominal. Este valor es válido solo para clusters de Mediciones.
- Pre TI: Tolerancia pre-inferior del valor nominal. Este valor es válido solo para clusters de Mediciones.
- Tol Inf: Tolerancia inferior del valor nominal. Este valor es válido solo para clusters de Mediciones.

3.2.3.3 ACQ Data Queue

- HWSyncs: Información de la muestras de transductores y encoders de cada Hardware que se utiliza para los cálculos de auxiliares y mediciones.
- HWASyncs: Información de la muestras de los correspondientes hardwares asíncronos.

4 METODOLOGÍA DE CÁLCULO

4.1 DESCRIPCIÓN

En el siguiente apartado se explicara cómo son verificados los errores de sintaxis, como así también, el método elegido para el cálculo, partiendo de la conversión a la notación algebraica RPN y terminado con el resultado pretendido por el programa.

4.2 VERIFICACIÓN DE SINTAXIS

4.2.1 Descripción

Consta de la comprobación de los términos ingresados por el usuario o aquellos que han sido previamente cargados, con la finalidad de que la sintaxis con la cual se proceda a trabajar sea la adecuada para el programa.

4.2.2 Chequeo de Sintaxis

El chequeo de sintaxis se realiza partiendo de la "Eliminación de espacios vacíos" y continuando por la "Conversión de todos los campos a Mayúscula". La razón de esta intervención es para que el proceso siguiente pueda ser más efectivo. Una vez concluido con el trabajo anterior se procede a buscar errores sobre la sintaxis. Aquí se verifica que la función este bien escrita respetando las formulas, variables validas, tipos de resultados de mediciones, tipos de resultados post-procesados y medidas post-procesadas que han sido solicitadas. También se incluye el análisis de repetición contigua de operadores y operandos, falta de paréntesis, etc. A continuación de este paso se divide completamente la función en constantes, variables, operandos y funciones, donde es posible analizar particularmente cada una de las expresiones que conforman la función, chequeando que no halla incoherencia en ninguna de estas. En el caso de que una función esté mal escrita se devolverá un mensaje de error indicando cual ha sido la falla y en qué posición se originó, a consecuencia de ello los cambios sobre esta misma no serán guardados por lo cual se deberá ingresar nuevamente la función. La figura 1 pone en evidencia un caso de error y su mensaje que emerge a causa del mismo.

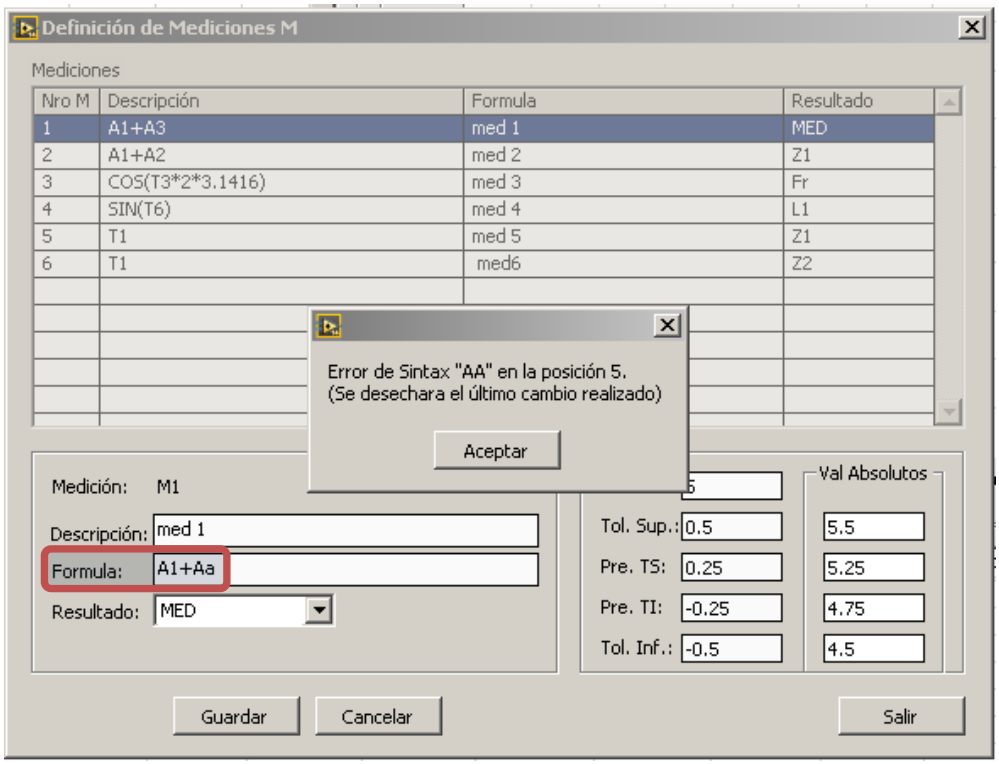


Figura 1. Caso de error en Sintaxis.

Por ultimo si la función pasa el test, se divide a toda la expresión matemática en partes llamadas “Tokens” los cuales son necesarios para los siguientes procesos. Aquí mismo también se realiza el armado de los arrays de variables y constantes que forman partes del cluster “Parsed Expression”.

4.2.3 Reglas de Sintaxis

Para el óptimo funcionamiento del sistema es sumamente necesario respetar las reglas de sintaxis. Para ello a continuación mostramos un listado de advertencias que debe tener el usuario al ingresar las expresiones matemáticas que se desean resolver.

1) No se admiten corchetes o llaves.
2) No permite el cálculo de otras funciones que no sean las solicitadas.
3) No admite el ingreso de otras variables o tipos de resultados que no sean los solicitados.
4) No permite la repetición contigua de operandos o variables.
5) No admite incoherencias matemáticas.

4.3 RPN (NOTACIÓN REVERSA POLACA)

4.3.1 Descripción

Su principio de funcionamiento consiste en evaluar los datos directamente cuando se introducen y manejarlos dentro de una estructura LIFO (Last In First Out), lo que optimiza los procesos a la hora de programar.

Este procedimiento es ventajoso ya que los cálculos se realizan secuencialmente según se van introduciendo operadores, en vez de tener que esperar a escribir la expresión al completo. Debido a esto, se cometen menos errores al procesar cálculos complejos. Además el proceso de apilación permite guardar resultados intermedios para un uso posterior. No requiere paréntesis ni reglas de preferencia, ya que el proceso de apilamiento permite calcular la expresión por etapas.

4.3.2 Conversión

Básicamente la diferencias con el método algebraico o notación de infijo es que, al evaluar los datos directamente al introducirlos, no es necesario ordenar la evaluación de los mismos, y que para ejecutar un comando, primero se deben introducir todos sus argumentos, así, para hacer una suma 'a+b=c' el RPN lo manejaría 'a b +', dejando el resultado 'c' directamente. Aquí lo que cambia es que el lugar donde va el operador. La Figura 2 detalla un simple ejemplo que pone en evidencia el procedimiento que se utilizó para el modulo Matemático.

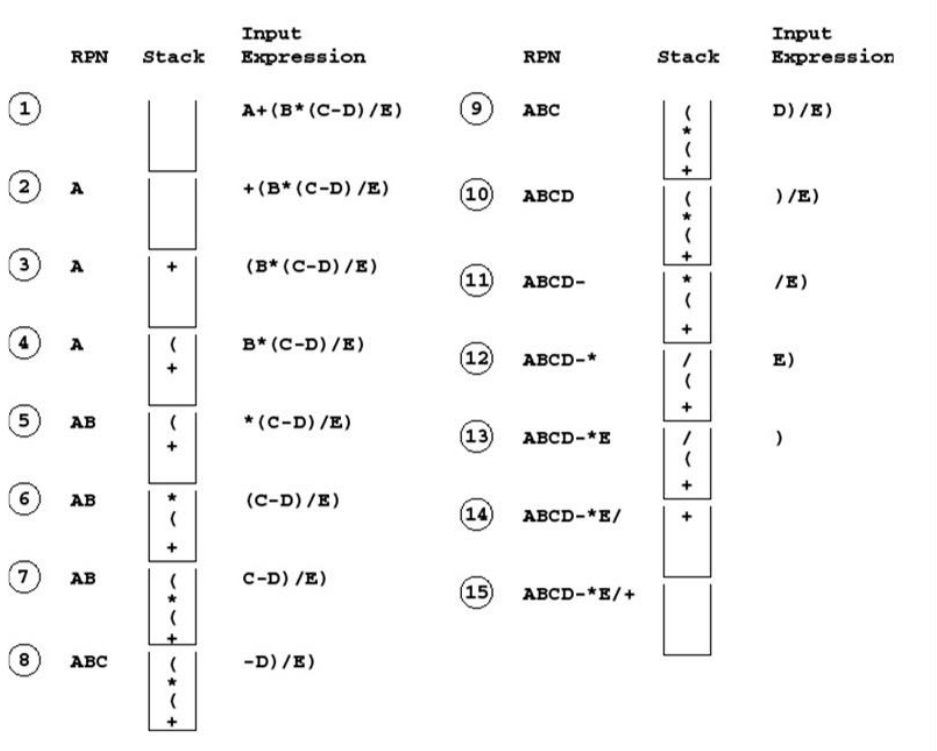


Figura 2. Conversión a Notación Inversa Polaca o RPN.

Como se pudo observar en la imagen anterior es necesario tener descompuesta toda la cadena de caracteres para poder ser procesadas, es por ello que el módulo de chequeo realiza esta acción así el bloque RPN solo tiene que proceder a realizar la conversión. El programa que se realizo es una copia fiel de este

método funcionando con un stack que almacena los operandos y un array que ordena las variables, constantes y operandos tal como la técnica lo propone.

4.4 CALCULO DE FUNCIONES

4.4.1 Descripción

El cálculo de funciones se realiza una vez que esta misma ha pasado por el chequeo de sintaxis y fue convertida a RPN. Este módulo comienza mapeando los valores de los transductores, continua resolviendo los cálculos pertenecientes a las medidas auxiliares y por ultimo calcula las medidas y sus resultados.

4.4.2 Mapeo

En la Figura 3 es posible visualizar el procedimiento de mapeo en el cual el usuario puede elegir que transductores desea utilizar (THW) del Hardware síncrono que desee y que nombre virtual le designara (TV).

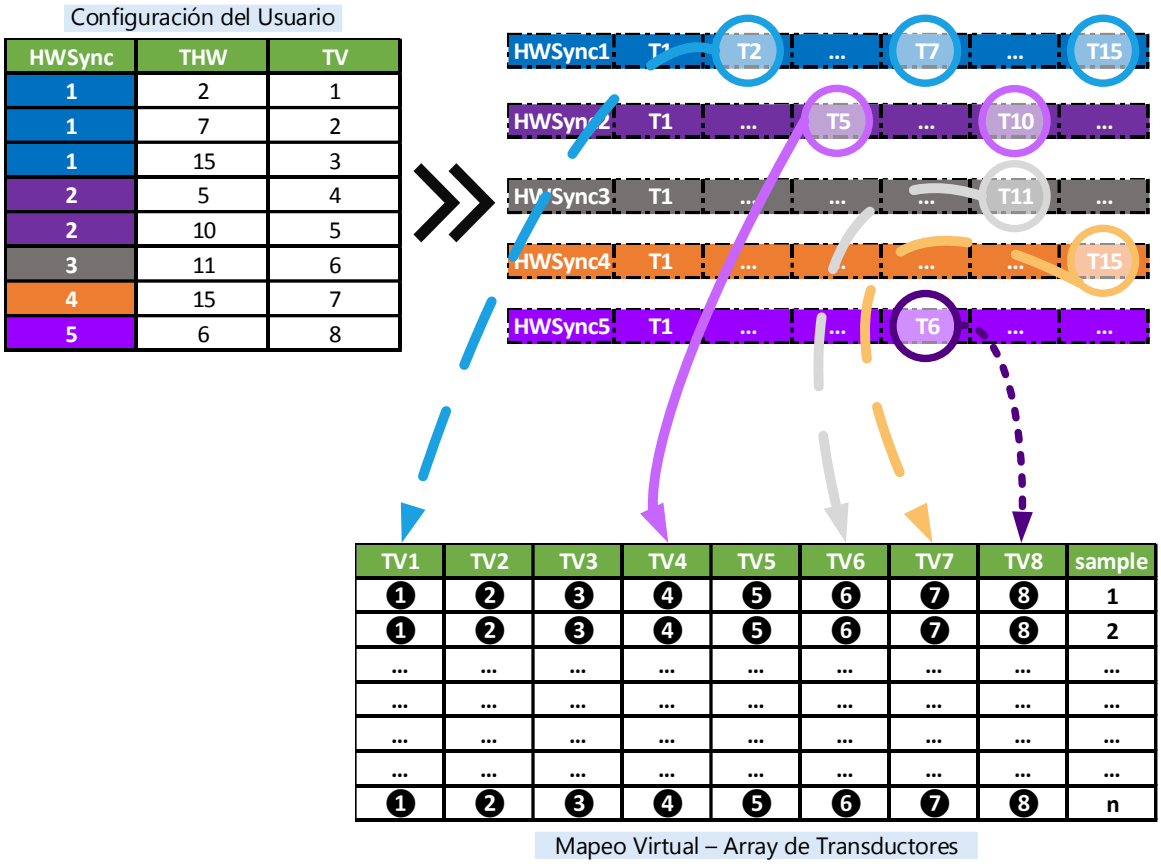


Figura 3. Mapeo de Transductores.

Este procedimiento es necesario para que el programa pueda resolver con más eficiencia los cálculos propuestos además de que permite al usuario poder independizarse de cada hardware y englobar a todos los transductores con su número de referencia.

4.4.3 Calculo de Auxiliares

El cálculo de auxiliares se basa en tomar la estructura que nos dejó el convertidor de RPN y con los valores de transductores mapeados, las constantes y las dimensiones ingresadas calcular el resultado final correspondiente a cada uno. En esta etapa solo se evalúan transductores, constantes y dimensiones.

4.4.4 Calculo de Mediciones y Resolución de Resultados

El cálculo de mediciones se ejecuta una vez que ha finalizado exitosamente el cálculo de auxiliares. Así una vez que se ejecuta este módulo tiene la capacidad de no solo poder utilizar los transductores, constantes y dimensiones sino que también puede usar las medidas auxiliares. Con la misma idea con la que se resolvieron las mediciones auxiliares se procede a resolver las mediciones, concluyendo así con un conjunto de resultados que serán utilizados por el bloque de resolución de resultados para obtener el valor final que se busca. La Figura 4 muestra en un simple diagrama de bloques como es el flujo de datos y como se comparte la información necesaria para el cálculo de cada uno de los bloques que constituyen el módulo de resolución de funciones matemáticas.

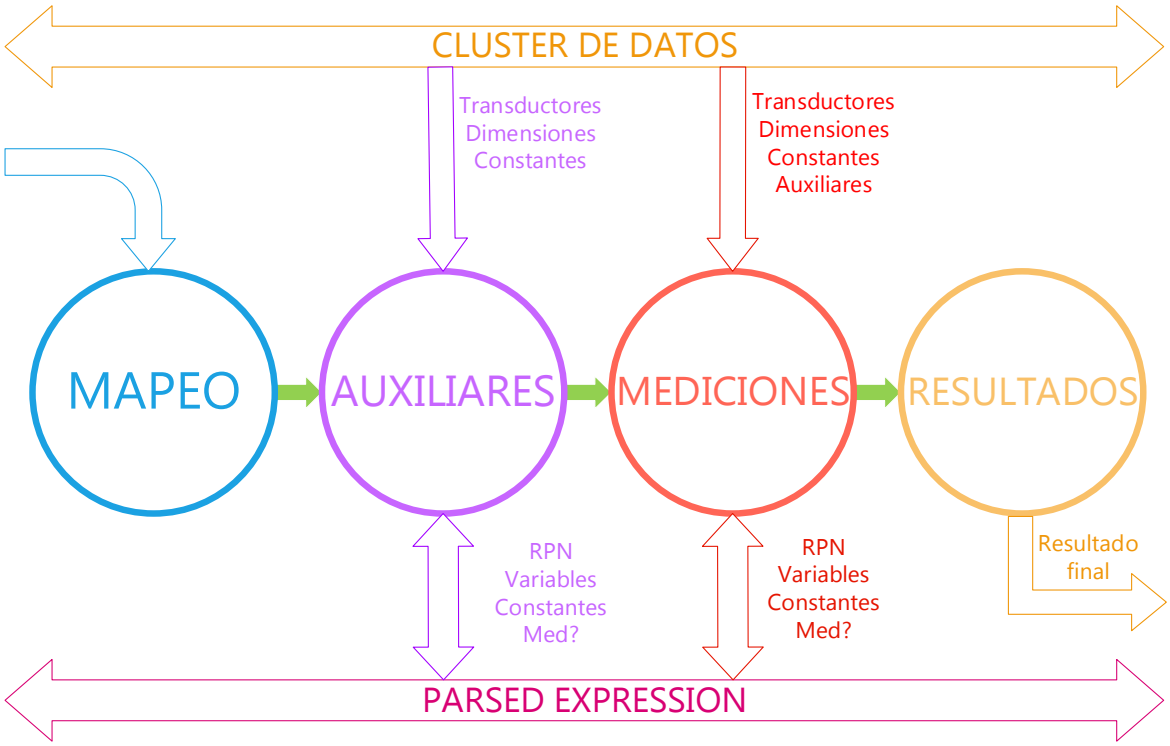


Figura 4. Calculo de Funciones.