App Note 2965: 1-Wire Master Device Configuration

The 'family-code' embedded in the lasered ROM number of each 1-Wire device signifies a specific device type. Since each device type has different features and commands, it is imperative the 1-Wire master knows how to translate this 'family-code' into the correct commands. This document presents a method to dynamically configure the 1-Wire master to correctly communicate with a previously unknown 1-Wire device type by providing the 1-Wire master with an XML configuration file. This document was originally created to support the IEEE 1451.4 A Smart Transducer Interface for Sensors and Actuators–Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats standards committee.

Introduction

The 'family-code' embedded in the lasered ROM number of each 1-Wire device signifies a specific device type. Since each device type has different features and commands, it is imperative the 1-Wire® master knows how to translate this 'family-code' into the correct commands. Unfortunately, since the 'family-code' is only an 8-bit value it is impossible to encode all of the features and commands in it. Instead the 1-Wire master must make this association by different means. One method is to hardcode this association in the source code of the 1-Wire master. It can then be updated by rewriting the source code to accommodate new devices. This method is expensive and in some cases impossible thus relegating some 1-Wire masters to only deal with legacy devices.

This document presents a method to dynamically configure the 1-Wire master to correctly communicate with a previously unknown 1-Wire device type by providing the 1-Wire master with a configuration file. The 1-Wire master could be updated with the latest 1-Wire devices by providing a new configuration file. This document describes the format of one such configuration file utilizing XML. This document was originally created to support the **IEEE 1451.4 A Smart Transducer Interface for Sensors and Actuators - Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats** standards committee.

The appendix proposes a method where a generic family code (for example FD hex) could be differentiated by means of a read-only configuration memory page on the device. No devices currently implement this method.

Command Notation

It is assumed that each 1-Wire master must come with the ability to search for 1-Wire devices and read the unique ROM number associated with each device. From the ROM number the 8-bit 'family-code' can be extracted. The 1-Wire master will then perform 1-Wire operations as defined by this configuration file based on the 'family-code'. By examining all 1-Wire device operations, a minimum set of commands was derived. The commands are described in Table 1 along with a suggested notation. Table 2 describes additional commands that add verification to the command sequences.

Table 1. Core 1-Wire Commands

NOTATION	COMMAND DESCRIPTION
XX	Send the following hex byte value to the 1-Wire bus. If this hex byte is within a CRC block then calculate the CRC on the result of the 1-Wire operation (see verification commands).
{L, delay}	Delay for 'L' milliseconds
{M}	Select the device with a 1-Wire reset, Match ROM command, and device ROM

{P}	Prime 1-Wire power delievery (strong pull-up) or to occur after the next 1-Wire byte
{N}	Restore normal pull-up
{U}	Issued a 12-volt pulse (used in EPROM programming)
{Ax}	Supply a memory address where 'x' is a number 0,1,...representing the LSbyte to MSbyte. For example '{A0}{A1}' would specify a 16-bit address with the least significant byte first followed by the most significant byte.
{Dx}	Data to write to a memory device where the 'x' is a number 0,1,...representing the LSbyte to MSbyte of the data. For example '{D0} {D1} {D2}' is three bytes of data to write. Note the master processing these commands would place the actual data into the command flow.
{R}	Read memory bytes to end of memory. All values read are valid data however now verification is performed.

Table 2. Verification Commands

NOTATION	COMMAND DESCRIPTION
{dx}	Data to read. This data can be for verification of data written to a memory 1-Wire device or result data such as a temperature conversion. Note it is in same format as the {Dx} command where 'x' is a number indicating the byte number with {d0} being the LSbyte.
{T}	Success is reading toggling bits such as 0xAA or 0x55.
{00}	Success is reading all 0's such as 0x00
{FF}	Success is reading all 1's such as 0xFF
{CRC16,start,seed}	Start CRC16 calculation by first setting the CRC16 to the provided 'seed' represented in hex notation. All following command bytes will be included in the calculation until the 'check' command is found.
{CRC16,check,value}	Check the CRC16 calculated value to make sure it equals the provided hex 'value'. If it is not then this is a failure. The CRC16 calculation can be stopped after the check.
{CRC8,start,seed}	Start CRC8 calculation by first setting the CRC8 to the provided 'seed' represented in hex notation. All following command bytes will be included in the calculation until the 'check' command is found.
{CRC16,check,value}	Check the CRC8 calculated value to make sure it equals the provided hex 'value'. If it is not then this is a failure. The CRC8 calculation can be stopped after the check.

See Figure 1 to see an example command sequence to read the scratchpad of the DS18B20.

Command Sequence
{M} BE {CRC8,start,0} {d0} {d1} FF FF FF FF FF FF FF {CRC8,check,0x00}

1-Wire Master Translation

{M}	Send 1-Wire reset, Match ROM command and ROM number
BE	Send 0xBE (Read ScratchPad) command
{CRC8,start,0x00}	Initialize the CRC8 to 0, begin CRC8 calculation
{d0} {d1}	Read the temperature by sending 0xFF 0xFF. Calculate CRC8
FF FF FF FF FF FF FF	Read rest of scratchpad. Calculate CRC8 on each byte result
{CRC8,check,0x00}	Check CRC8 value. If it is 0x00 then SUCCESS.

Figure 1. DS18B20 Read Temperature Command Sequence and 1-Wire Master Translation.

Device Types

The general types of 1-Wire devices covered by this document are Memory, Switch, and Temperature.
The Memory device has some kind of data storage memory area. It may be write-once but must support multiple reads. It is often arranged in pages and is usually written a page at a time. A Memory device may have multiple banks of

memory with different attributes.

The Switch device can control a latch. The latch may connect the output to ground (lowside) or to the communication channel (highside). Some Switch devices can also sense voltage. A Switch device can have multiple channels.

The Temperature device returns a temperature value in Celsius. The result is a signed value representing temperature units. The unit conversion to Celsius is provided.

Each device type contains one or more standard operations. For example each Temperature device has a 'read' operation. Table 3 shows the standard operations and attributes of each device type.

Table 3. Device Operations and Attributes by Type

DEVICE TYPE	OPERATIONS	ATTRIBUTES
Memory	Read Write	Read/Write/ReadOnly/WriteOnce Starting physical address Number of pages Page length in bytes
Switch	Read Latch Enable Latch Disable Latch Read Level (optional)	HIghSIde/LowSide
Temperature	Read	Min Temperature Max Temperature Step (unit of Celsius returned from Read)

The 'Setup' operation is also included in any of the device type descriptions. A 'Setup' is a command sequence that readies the device for operation. For any of the 'Read' operation there are also two attributes 'AndMask' and 'Polarity'. The 'AndMask' is a hex value that is bitwise anded with the result data described in the command sequence with {d0}. The 'Polarity' indicates that the operation is 'TRUE' if it matches the resulting value from the 'AndMask'. For example when reading the latch state (Read Latch) of a DS2406 channel A, the AndMask="0x01" and Polarity="0x00". So the value read from the command sequence is bitwise anded with 0x01 and if the result is 0, the latch is ON.

Configuration Format

The XML syntax was selected for the example configuration file format. Since XML is so 'eXtensible' it was easy to incorporate the device types, operations, attributes and the actual command sequences into a human readable format. The overall 'tag' for grouping these descriptions was . Within this group are individual device descriptions with a specified family code attribute, for example: . Each device group can contain , , or groups that correspond to the device types already described. Note that some devices may have more then one channel and group. For example the DS2406 has both memory and switch groups since it incorporates both of these features. See the Figure 2 in the appendix for an example XML file describing six different 1-Wire devices. Two of these devices are memory, two are switches, and two are temperature devices.

Examples

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- The device description file follows the schema defined in ??? and
defines devices DS2433, DS2430, DS2406,DS2409,DS18S20,DS1920 and DS18B20.-->
<DeviceDescriptions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="\Device Schema\Device Schema.xsd">
```

```
<DeviceDescriptions>
   <Device FamilyCode="0x23">
      <Description>
         DS2433, 4kbit EEPROM
      </Description>

      <MemoryBank attributes="ReadWrite">
      <Description>
         Main Memory
      </Description>

      <StartAddress> 0x0000 </StartAddress>
      <Pages> 16 </Pages>
      <PageLength> 32 </PageLength>

      <Write>
         <WriteScratchPad>
             {M} {CRC16,start,0}
             0F {A0} {A1}
             {D0} {D1} {D2} {D3} {D4} {D5} {D6} {D7}
             {D8} {D9} {D10} {D11} {D12} {D13} {D14} {D15}
             {D16} {D17} {D18} {D19} {D20} {D21} {D22} {D23}
             {D24} {D25} {D26} {D27} {D28} {D29} {D30} {D31}
             FF FF {CRC16,check,0xB001}
          </WriteScratchPad>
          <CopyScratchPad>
              {M} 55 {A0} {A1} {P} 1F {L,10} {N} {T}
          </CopyScratchPad>
        </Write>

        <Read>
           <ReadMemory>
               {M} F0 {A0} {A1} {R}
           </ReadMemory>
        </Read>

     </MemoryBank>
  </Device>

  <Device FamilyCode="0x14">
     <Description>
        DS2430A, 32-byte EEPROM with locking register
      </Description>

     <MemoryBank attributes="ReadWrite">
        <Description>
           Main Memory
        </Description>

        <StartAddress> 0x0000 </StartAddress>
        <Pages> 1 </Pages>
        <PageLength> 32 </PageLength>

        <Write>
           <WriteScratchPad>
               {M} 0F {A0}
               {D0} {D1} {D2} {D3} {D4} {D5} {D6} {D7}
```

```
            {D8} {D9} {D10} {D11} {D12} {D13} {D14} {D15}
            {D16} {D17} {D18} {D19} {D20} {D21} {D22} {D23}
            {D24} {D25} {D26} {D27} {D28} {D29} {D30} {D31}
        </WriteScratchPad>
        <ReadScratchPad>
            {M} AA {A0}
            {d0} {d1} {d2} {d3} {d4} {d5} {d6} {d7}
            {d8} {d9} {d10} {d11} {d12} {d13} {d14} {d15}
            {d16} {d17} {d18} {d19} {d20} {d21} {d22} {d23}
            {d24} {d25} {d26} {d27} {d28} {d29} {d30} {d31}
        </ReadScratchPad>
        <CopyScratchPad>
            {M} 55 {P} A5 {L,20} {N}
        </CopyScratchPad>
     </Write>

     <Read>
        <ReadMemory>
            {M} F0 {A0} {R}
        </ReadMemory>
     </Read>

  </MemoryBank>
  <MemoryBank attributes="WriteOnce">
     <Description>
        Application Register
     </Description>

     <StartAddress> 0x0000 </StartAddress>
     <Pages> 1 </Pages>
     <PageLength> 8 </PageLength>

     <Write>
        <WriteAppReg>
            {M} 99 {A0}
            {D0} {D1} {D2} {D3} {D4} {D5} {D6} {D7}
        </WriteAppReg>
        <ReadAppReg>
            {M} C3 {A0}
            {d0} {d1} {d2} {d3} {d4} {d5} {d6} {d7}
        </ReadAppReg>
        <CopyAndLock>
            {M} 5A {P} A5 {L,20} {N}
        </CopyAndLock>
     </Write>

     <Read>
        <ReadAppReg>
            {M} C3 {A0} {R}
        </ReadAppReg>
     </Read>
  </MemoryBank>
</Device>

<Device FamilyCode="0x12">
   <Description>
      DS2406, dual channel switch with 1kbit EPROM
```

```
</Description>

<MemoryBank attributes="WriteOnce">
   <Description>
      Main Memory
   </Description>

   <StartAddress> 0x0000 </StartAddress>
   <Pages> 4 </Pages>
   <PageLength> 32 </PageLength>

   <Write>
      <WriteScratchPad>
          {M} {CRC16,start,0}
          0F {A0} {A1} {D0}
          FF FF {CRC16,check,0xB001}
      </WriteScratchPad>
      <Program>
          {U}
      </Program>
      <ReadVerify>
          {d0}
      </ReadVerify>
   </Write>

   <Read>
      <ReadMemory>
          {M} F0 {A0} {A1} {R}
      </ReadMemory>
   </Read>

</MemoryBank>
<SwitchChannel attributes="LowSide">
   <Description>
      PIO-A
   </Description>

   <ReadLatch AndMask="0x01" Polarity="0x00">
      {M} {CRC16,start,0}
      F5 55 FF {d0}
      FF FF {CRC16,check,0xB001}
   </ReadLatch>

   <ReadLevel AndMask="0x04" Polarity="0x04">
      {M} {CRC16,start,0}
      F5 55 FF {d0}
      FF FF {CRC16,check,0xB001}
   </ReadLevel>

   <EnableLatch>
      {M} {CRC16,start,0}
      F5 05 FF 00
      FF FF {CRC16,check,0xB001}
   </EnableLatch>

   <DisableLatch>
      {M} {CRC16,start,0}
```

```
         F5 05 FF FF
         FF FF {CRC16,check,0xB001}
      </DisableLatch>

   </SwitchChannel>

   <SwitchChannel attributes="LowSide">
      <Description>
         PIO-B
      </Description>

      <ReadLatch AndMask="0x02" Polarity=•g0x00•h>
         {M} {CRC16,start,0}
         F5 55 FF {d0}
         FF FF {CRC16,check,0xB001}
      </ReadLatch>

      <ReadLevel AndMask="0x08" Polarity="0x08">
         {M} {CRC16,start,0}
         F5 55 FF {d0}
         FF FF {CRC16,check,0xB001}
      </ReadLevel>

      <EnableLatch>
         {M} {CRC16,start,0}
         F5 09 FF 00
         FF FF {CRC16,check,0xB001}
      </EnableLatch>

      <DisableLatch>
         {M} {CRC16,start,0}
         F5 09 FF FF
         FF FF {CRC16,check,0xB001}
      </DisableLatch>

   </SwitchChannel>
</Device>

<Device FamilyCode="0x1F">
   <Description>
      DS2409, 1-Wire Coupler
   </Description>

   <SwitchChannel attributes="HighSide">
      <Description>
         Main
      </Description>

      <ReadLatch AndMask="0x01" Polarity="0x00">
         {M} 5A 18 {d0}
      </ReadLatch>

      <ReadLevel AndMask="0x02" Polarity="0x02">
         {M} 5A 18 {d0}
      </ReadLevel>

      <ReadActivity AndMask="0x10" Polarity="0x10">
         {M} 5A 18 {d0}
```

```
      </ReadActivity>

      <EnableLatch>
         {M} A5 FF
      </EnableLatch>

      <DisableLatch>
         {M} 66 FF
      </DisableLatch>
   </SwitchChannel>

   <SwitchChannel attributes="HighSide">
      <Description>
         Auxilary
      </Description>

      <ReadLatch AndMask="0x04" Polarity=•g0x00•h>
         {M} 5A 18 {d0}
      </ReadLatch>

      <ReadLevel AndMask="0x08" Polarity="0x08">
         {M} 5A 18 {d0}
      </ReadLevel>

      <EnableLatch>
         {M} 33 FF FF FF
      </EnableLatch>

      <DisableLatch>
         {M} 66 FF
      </DisableLatch>
    </SwitchChannel>
</Device>

<Device FamilyCode="0x10">
   <Description>
      DS18S20/DS1920, fixed resolution temperature
   </Description>

   <TemperatureChannel min="-55" max="125" step="0.5">
      <Read>
         <Recall>
            {M} B8
         </Recall>
         <Conversion>
            {M} {P} 44 {L,750} {N} {FF}
         </Conversion>
         <Result>
            {M} BE {CRC8,start,0} {d0} {d1}
            FF FF FF FF FF FF FF {CRC8,check,0x00}
         </Result>
      </Read>
   </TemperatureChannel>
</Device>

<Device FamilyCode="0x28">
   <Description>
```

```
         DS18B20, high-resolution temperature
      </Description>

      <TemperatureChannel min="-55" max="125" step="0.0625">
         <Setup>
            <WriteScatchPad>
               {M} 00 00 7F
            </WriteScatchPad>

            <CopyScatchPad>
               {M} {P} 48 {L,10} {N}
            </CopyScatchPad>
        </Setup>

        <Read>
           <Recall>
              {M} B8
           </Recall>
           <Conversion>
              {M} {P} 44 {L,750} {N} {FF}
           </Conversion>
           <Result>
              {M} BE {CRC8,start,0} {d0} {d1}
              FF FF FF FF FF FF FF {CRC8,check,0x00}
           </Result>
        </Read>
     </TemperatureChannel>
  </Device></DeviceDescriptions>
```

Figure 2. Example XML Configuration File for Six 1-Wire Devices

XML Device Description Schema

The device description schema provides a template to add support for new devices to their systems. The schema defines devices that support memory, switching, and temperature reading.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- The IEEE 1451.4 XML device description schema provides a template for manufacturers
and
users of the IEEE14514 to add support for new devices to their systems. The schema
defines
devices that support memory, switching and temperature reading. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
      <xs:element name="Conversion" type="xs:string"/>
      <xs:element name="CopyAndLock" type="xs:string"/>
      <xs:element name="CopyScatchPad" type="xs:string"/>
      <xs:element name="CopyScratchPad" type="xs:string"/>
      <xs:element name="Description" type="xs:string"/>
      <xs:complexType name="IEEE1451_Dot4DeviceType">
            <xs:sequence>
                  <xs:element ref="Description"/>
                  <xs:element name="MemoryBank" type="MemoryBankType" minOccurs="0"
maxOccurs="unbounded"/>
                  <xs:element name="SwitchChannel" type="SwitchChannelType"
minOccurs="0"
```

```
maxOccurs="unbounded"/>
                    <xs:element name="TemperatureChannel" type="TemperatureChannelType"
minOccurs="0"/>
             </xs:sequence>
             <xs:attribute name="FamilyCode" use="required">
                    <xs:simpleType>
                           <xs:restriction base="xs:NMTOKEN">
                                  <xs:enumeration value="0x10"/>
                                  <xs:enumeration value="0x12"/>
                                  <xs:enumeration value="0x14"/>
                                  <xs:enumeration value="0x1F"/>
                                  <xs:enumeration value="0x23"/>
                                  <xs:enumeration value="0x28"/>
                           </xs:restriction>
                    </xs:simpleType>
             </xs:attribute>
      </xs:complexType>
      <xs:element name="DeviceDescriptions">
             <xs:complexType>
                    <xs:sequence>
                            <xs:element name="Device" type="IEEE1451_Dot4DeviceType"
maxOccurs="unbounded"/>
                    </xs:sequence>
             </xs:complexType>
      </xs:element>
      <xs:element name="DisableLatch" type="xs:string"/>
      <xs:element name="EnableLatch" type="xs:string"/>
      <xs:complexType name="MemoryBankType">
             <xs:sequence>
                    <xs:element ref="Description"/>
                    <xs:element ref="StartAddress"/>
                    <xs:element ref="Pages"/>
                    <xs:element ref="PageLength"/>
                    <xs:element name="Write" type="WriteType"/>
                    <xs:element name="Read" type="ReadType"/>
                    <xs:element name="CRCInformation" minOccurs="0">
                           <xs:complexType>
                                  <xs:sequence maxOccurs="unbounded">
                                         <xs:element name="CRCStartBitPageLocation"
type="xs:unsignedLong"/>
                                         <xs:element name="CRCBitLength"
type="xs:unsignedLong"/>
                                  </xs:sequence>
                           </xs:complexType>
                    </xs:element>
             </xs:sequence>
             <xs:attribute name="attributes" use="required">
                    <xs:simpleType>
                           <xs:restriction base="xs:NMTOKEN">
                                  <xs:enumeration value="ReadWrite"/>
                                  <xs:enumeration value="WriteOnce"/>
                           </xs:restriction>
                    </xs:simpleType>
             </xs:attribute>
      </xs:complexType>
      <xs:element name="PageLength" type="xs:unsignedLong"/>
```

```
	<xs:element name="Pages" type="xs:unsignedLong"/>
	<xs:element name="Program" type="xs:string"/>
	<xs:complexType name="ReadType">
		<xs:sequence>
			<xs:element ref="ReadMemory" minOccurs="0"/>
			<xs:element ref="ReadAppReg" minOccurs="0"/>
			<xs:element ref="Recall" minOccurs="0"/>
			<xs:element ref="Conversion" minOccurs="0"/>
			<xs:element ref="Result" minOccurs="0"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="ReadActivityType">
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute name="AndMask" type="xs:string"
use="required"/>
				<xs:attribute name="Polarity" type="xs:string"
use="required"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:element name="ReadAppReg" type="xs:string"/>
	<xs:complexType name="ReadLatchType">
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute name="AndMask" use="required">
					<xs:simpleType>
						<xs:restriction base="xs:NMTOKEN">
							<xs:enumeration value="0x01"/>
							<xs:enumeration value="0x02"/>
							<xs:enumeration value="0x04"/>
						</xs:restriction>
					</xs:simpleType>
				</xs:attribute>
				<xs:attribute name="Polarity" type="xs:decimal"
use="required"/>
			</xs:extension>
		</xs:simpleContent>
	</xs:complexType>
	<xs:complexType name="ReadLevelType">
		<xs:simpleContent>
			<xs:extension base="xs:string">
				<xs:attribute name="AndMask" use="required">
					<xs:simpleType>
						<xs:restriction base="xs:NMTOKEN">
							<xs:enumeration value="0x02"/>
							<xs:enumeration value="0x04"/>
							<xs:enumeration value="0x08"/>
						</xs:restriction>
					</xs:simpleType>
				</xs:attribute>
				<xs:attribute name="Polarity" use="required">
					<xs:simpleType>
						<xs:restriction base="xs:NMTOKEN">
							<xs:enumeration value="0x02"/>
							<xs:enumeration value="0x04"/>
```

```
                                        <xs:enumeration value="0x08"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>
        <xs:element name="ReadMemory" type="xs:string"/>
        <xs:element name="ReadScratchPad" type="xs:string"/>
        <xs:element name="ReadVerify" type="xs:string"/>
        <xs:element name="Recall" type="xs:string"/>
        <xs:element name="Result" type="xs:string"/>
        <xs:complexType name="SetupType">
                <xs:sequence>
                        <xs:element ref="WriteScatchPad"/>
                        <xs:element ref="CopyScatchPad"/>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="StartAddress" type="xs:string"/>
        <xs:complexType name="SwitchChannelType">
                <xs:sequence>
                        <xs:element ref="Description"/>
                        <xs:element name="ReadLatch" type="ReadLatchType"/>
                        <xs:element name="ReadLevel" type="ReadLevelType"/>
                        <xs:element name="ReadActivity" type="ReadActivityType"
minOccurs="0"/>
                        <xs:element ref="EnableLatch"/>
                        <xs:element ref="DisableLatch"/>
                </xs:sequence>
                <xs:attribute name="attributes" use="required">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="HighSide"/>
                                        <xs:enumeration value="LowSide"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
        </xs:complexType>
        <xs:complexType name="TemperatureChannelType">
                <xs:sequence>
                        <xs:element name="Setup" type="SetupType" minOccurs="0"/>
                        <xs:element name="Read" type="ReadType"/>
                </xs:sequence>
                <xs:attribute name="min" type="xs:byte" use="required"/>
                <xs:attribute name="max" type="xs:byte" use="required"/>
                <xs:attribute name="step" use="required">
                        <xs:simpleType>
                                <xs:restriction base="xs:NMTOKEN">
                                        <xs:enumeration value="0.0625"/>
                                        <xs:enumeration value="0.5"/>
                                </xs:restriction>
                        </xs:simpleType>
                </xs:attribute>
        </xs:complexType>
        <xs:complexType name="WriteType">
                <xs:sequence>
```

```
                    <xs:element ref="WriteScratchPad" minOccurs="0"/>
                    <xs:element ref="ReadScratchPad" minOccurs="0"/>
                    <xs:element ref="CopyScratchPad" minOccurs="0"/>
                    <xs:element ref="WriteAppReg" minOccurs="0"/>
                    <xs:element ref="ReadAppReg" minOccurs="0"/>
                    <xs:element ref="CopyAndLock" minOccurs="0"/>
                    <xs:element ref="Program" minOccurs="0"/>
                    <xs:element ref="ReadVerify" minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <xs:element name="WriteAppReg" type="xs:string"/>
        <xs:element name="WriteScatchPad" type="xs:string"/>
        <xs:element name="WriteScratchPad" type="xs:string"/>
</xs:schema>
```

Figure 2. XML Device Description Schema.

Appendix

Memory Configuration Page

The following general memory description describes an idealized memory device with a configuration page that provides all of the necessary information to utilize the remaining memory space. The configuration page could provide the device type differentiation that is currently implemented with the ROM family code but with more information conveyed. A common generic family code (for example FD hex) could be used for all devices with this configuration page.

All 1-Wire memory devices support the Read Memory command (F0 hex), and with the exception of the DS2430A, it requires 2 address bytes. For this example the Read Memory command will be used to retrieve the configuration page information at a fixed address of FF7F hex. The memory location will have a length byte, 26 bytes of data followed by an inverted CRC16 for validation. Table A1 provides the bit-level details of the configuration page format.

Table A1. Configuration Page Format

BYTE OFFSET	NAME	CONTENT	
0	Length	Length of data in the configuration page (fixed at 26)	
1	General_Flags	Bit 0	Memory type (1 EEPROM, 0 EEPROM)
		Bit 1	Scratchpad erased on read-memory (1 YES, 0 NO)
		Bit 2	Device has read page with CRC16 (1 YES, 0 NO)
		Bit 3	Device has write-once mode like pseudo EPROM (1 YES, 0 NO) (EEPROM only)
		Bit 4	Device has map of used pages (1 YES, 0 NO)
		Bit 5	not used 0
		Bit 6	not used 0
		Bit 7	not used 0
2	WriteProt_Flags	Bit 0	Individual page write-protect (1 YES, 0 NO)
		Bit 1	Global device write-protect (1 YES, 0 NO)
		Bit 2	Write protect register is organized with one page per bit (1 YES, 0 NO). If not then one page per byte

		Bit 3	not used 0
		Bit 4	not used 0
		Bit 5	not used 0
		Bit 6	not used 0
		Bit 7	not used 0
3	CRC_Flags	Bit 0	Write scratchpad has CRC16 (1 YES, 0 NO)
		Bit 1	Read scratchpad has CRC16 (1 YES, 0 NO)
		Bit 2	Read special memory command has CRC16 (1 YES, 0 NO)
		Bit 3	not used 0
		Bit 4	not used 0
		Bit 5	not used 0
		Bit 6	not used 0
		Bit 7	not used 0
4	Scratchpad_Length	Length of scratchpad in bytes (EEPROM only)	
5	Page_Length	Length of normal memory page in bytes	
6	Pages	Number of pages (2 bytes)	
8	Special_Pages	Number of special function pages	
9	Special_Page_Length	Length of special memory page in bytes	
10	ReadScratch_CMD	Read scratchpaf command	
11	Write_CMD	Write command (scratchpad for EEPROM)	
12	CopyScratch_CMD	Copy scratchpad command	
13	ReadPageCRC_CMC	Read page of memory with CRC16 command	
14	ReadSpecial_CMD	Read special memory page command	
15	Write_Special_CMD	Write special memory command	
16	WriteProt_Addr	Address of write-protect registers in special memory. (2 bytes)	
18	WriteProtDev_Addr	Address of write-protect entire device register in special memory. (2 bytes)	
20	WriteOnce_Addr	Address to write-once mode (pseudo EPROM) flag in special memory. (2 bytes)	
22	UsedPgs_Addr	Address in special memory for map of used pages. (2 bytes)	
24	UsedPgs_Offset	Bit offset of the map of used pages	
25	WriteProt_Value	Value written to special memory register to write-protect a page.	
26	WriteOnce_Value	Value written to special memory register to make a page write-once like pseudo EPROM.	
27	CRC16	Bitwise inverted CRC16 of bytes 0 to 24, LSByte first. (2 bytes)	

The following table lists the operations that will be described by the configuration page.

Table A2. Operations

OPERATION	EEPROM	EPROM	DESCRIPTION
Read Memory	X	X	Read memory with device generated CRC
Read Page with CRC	x	x	Read a page of memory with device generated CRC
Write Scratchpad	X		Write the scratchpad in preparation of writing ot memory

Read Scratchpad	X		Read the scratchpad to verify the write was correct
Copy Scratchpad	X		Copy the scratchpad to the final memory location
Write Memory		X	Write a byte to memory
Read speical page with CRC		x	Read a page of special memory with device generated CRC
Write special byte		x	Write a byte to the special memory
Write protect page	x	x	Write protect a page
Set page for write-once	x		Set an EEPROM page to be write-once like (pseudo EPROM)
Calculate Free Pages		x	Calculate the number of free pages in an EPROM device by looking at the map of used pages.

X supported by all devices of this type
x supported by some devices of this type
<blank> generally not supported by devices of this type

Operations Detail

The operations detail listed in Table A2 can be implemented with the details provided by the configuration page. This section provides the sequence and data fields to use to implement the operations.

Read Memory

- 1-Wire reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write ReadMemory command (F0 hex)
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Read data

Read Page with CRC
- If General_Flags.Bit2 = 1

1-Wire reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write ReadPageCRC_CMD
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Read Page_Length bytes (unless address is not at page beginning)
- Read bitwise inverted CRC16

Write Scratchpad
- If General_Flags.Bit0 = 1

1-Wire reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write Write_CMD
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Write data bytes

- If CRCFlags.Bit0 = 1 AND at end of page
    - Read bitwise inverted CRC16

Read Scratchpad

- If General_Flags.Bit0 = 1
  1-Wire reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write ReadScratch_CMD
- Read first address byte TA1, LSByte
- Read second address byte TA2, MSByte
- Read offest and status flags ES
- Read data bytes
- If CRCFlags.Bit1 = 1 AND at end of page
    - Read bitwise inverted CRC16

Copy Scratchpad

- If General_Flags.Bit0 = 1
  1-Wire reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write CopyScratch_CMD
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Write offset and status flags ES
- Strong pullup applied to 1-Wire for a minimum of 10ms
- Read confirmation byte (should be AA or 55)

Write Memory

- If General_Flags.Bit0 = 0
  1-Wire reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write Write_CMD
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Write data byte to write
- Read bitwise inverted CRC16 of command, address, and data (first pass) or address and data (second pass)
- Apply 480 µs 12V programming pulse on the 1-Wire
- Read confirmation data byte (should OR of old data and current data bytes)
- If next address to write is sequential then can send the next data byte...

Read Special Page with CRC

- If General_Flags.Bit2 = 1
  1-Wire reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write ReadSpecial_CMD
- Read first address byte TA1, LSByte
- Read second address byte TA2, MSByte

- Read Special_Page_Length bytes (unless address is not at page beginning)
- Read bitwise inverted CRC16

Write Special Byte

- If General_Flags.Bit0 = 0
  1-Wire reset and presence
- ROM level command sequence (read/search/match/overdrive match/overdrive skip)
- Write Write_Special_CMD
- Write first address byte TA1, LSByte
- Write second address byte TA2, MSByte
- Write data byte to write
- Read bitwise inverted CRC16 of command, address, and data (first pass) or address and data (second pass).
  Apply 480 µs 12V programming pulse on the 1-Wire
- Read confirmation data byte (should OR of old data and current data bytes)
- If next address to write is sequential then can send the next data byte...

Write Protect Page

- If WriteProt_Flags.Bit0 = 1
  If WriteProt_Flags.Bit2 = 1
    - Address = WriteProt_Addr + Page / 8
    - Data = WriteProt_Value Rotate left Remainder (Page / 8)
- Else
    - Address = WriteProt_Addr + Page
    - Data = WriteProt_Value
- Write Special Byte with Address and Data.

Set Page for Write-Once

- If General_Flags.Bit3 = 1
  Address = WriteOnce_Addr
- Data = WriteOnce_Value
- Write Special Byte with Address and Data

Mark Page Used

- If General_Flags.Bit4 = 1
  Address = UsedPgs_Addr + (Page + UsedPgs_Offset)/ 8
- Data = BitInverse (1 Rotate left Remainder ((Page + UsedPgs_Offest)/ 8)
- Write special byte at Address and Data

Calculate Free Pages

- If General_Flags.Bit4 = 1
  Address = UsedPgs_Addr
- Read special page with CRC starting at Address until (Special_Pages / 8) number of bytes read
- Count the number of 1's in the bytes read, this is the number of free pages

Table A3 provides example configuration pages using existing devices as a template. Note however, these devices do not currently contain the configuration page.

Table A3. Example Configuration Pages

Key
X single numbers are binary (0 or 1)
XX double numbers are in hex

#	Name	Content		DS2433	DS2406	DS2505	DS2506	DS2431 (future product)	DS28E04 (future product)
0	Length	Length of data in the configuration page		1A	1A	1A	1A	1A	1A
1	General_Flags	Bit 0	Memory type (1 EEPROM, 0 EPROM)	1	0	0	0	1	1
		Bit 1	Scratchpad erased on read-memory (1 YES, 0 NO)	1	0	0	0	1	1
		Bit 2	Device has read page with CRC16 (1 YES, 0 NO)	0	1	1	1	1	1
		Bit 3	Device has write-once mode like pseudo EPROM (1 YES, 0 NO) (EEPROM only)	0	0	0	0	1	1
		Bit 4	Device has map of used pages (1 YES, 0 NO)	0	1	1	1	0	0
		Bit 5	not used 0	0	0	0	0	0	0
		Bit 6	not used 0	0	0	0	0	0	0
		Bit 7	not used 0	0	0	0	0	0	0
2	WriteProt_Flags	Bit 0	Individual page write-protect (1 YES, 0 NO)	0	1	1	1	1	1
		Bit 1	Global device write-protect (1 YES, 0 NO)	0	0	0	0	0	0
		Bit 2	Write protect register is organized with one page per bit (1 YES, 0 NO). If no then is one page per byte.	0	1	1	1	0	0
		Bit 3	not used 0	0	0	0	0	0	0
		Bit 4	not used 0	0	0	0	0	0	0
		Bit 5	not used 0	0	0	0	0	0	0
		Bit 6	not used 0	0	0	0	0	0	0
		Bit 7	not used 0	0	0	0	0	0	0
3	CRC_Flags	Bit 0	Write scratchpad has CRC16 (1 YES, 0 NO)	1	0	0	0	1	1
		Bit 1	Read scratchpad has CRC16 (1 YES, 0 NO)	0	0	0	0	1	1
		Bit 2	Read special memory command has CRC16 (1 YES, 0 NO)	0	1	1	1	0	0
		Bit 3	not used 0	0	0	0	0	0	0
		Bit 4	not used 0	0	0	0	0	0	0
		Bit 5	not used 0	0	0	0	0	0	0
		Bit 6	not used 0	0	0	0	0	0	0
		Bit 7	not used 0	0	0	0	0	0	0
4	Scratchpad_Length	Length of scratchpad in bytes (EEPROM only)		20	00	00	00	08	20
5	Page_Length	Length of normal memory page in bytes		20	20	20	20	20	20
6	Pages	Number of pages		10 00	04 00	40 00	00 01	04 00	10 00
8	Special_Pages	Number of special function pages		00	01	0B	0B	01	02
9	Special_Page_Length	Length of special memory page in bytes		00	08	08	08	08	20
10	ReadScratch_CMD	Read scratchpad command		AA	00	00	00	AA	AA
11	Write_CMD	Write command (scratchpad for EEPROM)		0F	0F	0F	0F	0F	0F
12	CopyScratch_CMD	Copy scratchpad command		55	00	00	00	55	55
13	ReadPageCRC_CMD	Read page of memory with CRC16 command		00	A5	A5	A5	00	00
14	ReadSpecial_CMD	Read special memory page command		00	AA	AA	AA	F0	F0
15	Write_Special_CMD	Write special memory command		00	55	55	55	0F	0F
16	WriteProt_Addr	Address of write-protect registers in special memory. (2 bytes)		00 00	00 00	00 00	00 00	80 00	00 20
18	WriteProtDev_Addr	Address of write-protect entire device register in special memory. (2 bytes)		00 00	00 00	00 00	00 00	00 00	00 00
20	WriteOnce_Addr	Address to write-once mode (pseudo EPROM) flag in special memory. (2 bytes)		00 00	00 00	00 00	00 00	80 00	00 20
22	UsedPgs_Addr	Address in special memory for map of used pages. (2 bytes)		00 00	00 00	40 00	40 00	00 00	00 00

20	WriteOnce_Addr	Address to write-once node (pseudo EPROM) flag in special memory. (2 bytes)	00 00	00 00	00 00	00 00	80 00	00 20
22	UsedPgs_Addr	Address in special memory for map of used pages. (2 bytes)	00 00	00 00	40 00	40 00	00 00	00 00
24	UsedPgs_Offset	Bit offset of the map of used pages	00	04	00	00	00	00
25	WriteProt_Value	Value written to special memory register to write-protect a page	00	00	00	00	55	55
26	WriteOnce_Value	Value written to special memory register to make a page write-once like pseudo EPROM.	00	00	00	00	AA	AA
27	CRC16	Bitwise inverted CRC16 of bytes 0 to 24, LSByte first. (2 bytes)	xx xx	xx xx	xx xx	xx xx	xx xx	xx xx

More Information

DS2430A: QuickView -- Full (PDF) Data Sheet -- Free Samples