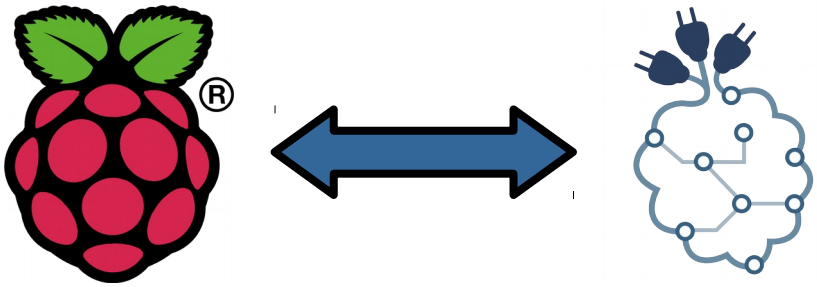
Application Note

Prozessdaten

Datenaustausch zwischen PiXtend-Controller und Raspberry Pi verstehen und nutzen



Stand 24.10.2016, V1.04

Qube Solutions UG (haftungsbeschränkt)
Arbachtalstr. 6, 72800 Eningen, Deutschland
http://www.qube-solutions.de/
http://www.pixtend.de

Inhaltsverzeichnis

1. Einleitung & Allgemeines

In dieser App-Note möchten wir Ihnen erläutern, welche Prozessdaten zwischen PiXtend-Mikrocontroller und dem Raspberry Pi ausgetauscht werden. Wir sprechen hier von Prozessdaten, da es um die Daten von Ein- und Ausgängen des PiXtend-Boards geht. Die Control- und Status-Bytes werden in einer separaten App-Note erklärt.

Die Informationen im vorliegenden Dokument sind besonders für Personen gedacht, die sich näher mit der Funktion und der Weiterentwicklung von PiXtend beschäftigen möchten.

Bei den PiXtend-Beispielprogrammen für *CODESYS* werden die Daten bereits korrekt übertragen, ausgewertet und ggf. aufbereitet.

Bei den Linux-Tools und der Verwendung der PiXtend-Library ist es wichtig wie die Daten beeinflusst und ausgewertet werden können.

Sollten Fragen offenbleiben, so bitten wir Sie uns per E-Mail (support@pixtend.de) in Kenntnis zu setzen. Sie erhalten schnellst möglich eine Antwort und weitere Informationen.

Diese Application-Note ist gleichermaßen für PiXtend V1.2 und V1.3 gültig.

Die jeweils neusten Versionen aller Dokumente und Software-Komponenten finden Sie im Download-Bereich unserer Homepage: http://www.pixtend.de/downloads/

1.1 Prozessdaten im Überblick

Es wird zwischen zwei Arten von Prozessdaten unterschieden:

1. **Prozessdaten die vom Raspberry Pi an PiXtend übertragen werden**

   Daten für digitale Ausgänge, Relais und GPIOs (als Ausgänge)
   PWM-Daten

- DIGITAL_OUT
- RELAYS
- GPIO_OUT
- PWMX (L/H)

2. **Prozessdaten die von PiXtend an den Raspberry Pi übertragen werden**

   Daten der digitalen und analogen Eingänge, GPIOs (als Eingänge)
   Temperatur und Luftfeuchtigkeit von DHT11/22- bzw. AM2302-Sensoren

- DIGITAL_IN
- ANALOG_INX (L/H)
- GPIO_IN
- TEMPX (L/H)
- HUMIDITY (L/H)
- DIGITAL_OUT_READ *(verfügbar ab Mikrocontroller V12.6 bzw. V13.2)*
- RELAYS_READ *(verfügbar ab Mikrocontroller V12.6 bzw. V13.2)*

Die analogen Ausgänge werden nicht über den PiXtend-Mikrocontroller angesteuert und sind in diesem Dokument nicht genauer erläutert.

2. Beschreibung der Prozessdaten

In diesem Kapitel möchten wir Ihnen die Prozessdaten genau erläutern. Es wird jedes Byte und jedes relevante Bit genau beleuchtet.

Die beschriebenen Daten gelten sowohl für den Manual-Mode als auch für den Automatic-Mode.

Falls Ihnen die Prozessdaten nicht zusagen sollten, so bleibt die Möglichkeit offen, die Firmware des PiXtend-Controllers auf Ihre Bedürfnisse anzupassen. Die Quelltexte finden Sie auf unserer Homepage im Download-Bereich.

Beim Umgang und besonders beim Experimentieren mit den Prozessdaten ist Vorsicht geboten. Angeschlossene Sensoren und Aktoren können bei falscher Handhabung undefinierte Zustände einnehmen bzw. falsche Werte ausgeben.

Wird mit PiXtend eine Maschine, ein Gerät oder Prozess gesteuert oder geregelt, können so gefährliche Zustände eintreten. Machen Sie sich mögliche Gefahrenquellen frühzeitig bewusst.

Trennen Sie im Zweifelsfall die Verbindungen zu den Geräten, Sensoren, Motoren usw. von PiXtend und der Spannungsversorgung ab, um Gefahren für Mensch und Maschine zu minimieren.

Auf die Prozessdaten kann per *CODESYS*, dem *pixtendtool* und *pxauto* zugegriffen werden. Auch Ihre eigenen Programme können die Daten natürlich beeinflussen und verarbeiten. Wir empfehlen in diesem Fall die Verwendung der PiXtend-C-Library (pxdev).

2.1 DIGITAL_OUT

Bit	7	6	5	4	3	2	1	0
Name	**-**	**-**	**DO5**	**DO4**	**DO3**	**DO2**	**DO1**	**DO0**
Startwert	0	0	0	0	0	0	0	0

Bit 0..5

Die sechs digitalen Ausgänge sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von DO0. Bit 7 (MSB) und Bit 6 sind nicht belegt.

Als Startwert nach dem Power-Up bzw. Reset des Mikrocontrollers ist für das gesamte DIGITAL_OUT-Byte „0“. Die Ausgänge sind im Start-/Ruhezustand deaktiviert.

Durch schreiben einer „1“ in eines der Bits (0..5) wird der entsprechende Ausgang aktiviert. Die LED auf dem PiXtend-Board leuchtet auf.

Um die Funktion der Ausgänge zu testen, kann für das DIGITAL_OUT-Byte der Wert 255 (dezimal) bzw. 0xFF (hex) geschrieben werden. Alle sechs Ausgänge werden aktiviert. Dass dabei auch die Bits 6 und 7 gesetzt werden, hat keine Auswirkung.

2.2 RELAYS

Bit	7	6	5	4	3	2	1	0
Name	**-**	**-**	**-**	**-**	**RELAY3**	**RELAY2**	**RELAY1**	**RELAY0**
Startwert	0	0	0	0	0	0	0	0

Bit 0..3

Die vier Relais-Ausgänge sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von RELAY0. Bit 7 (MSB) bis Bit 4 sind nicht belegt.

Als Startwert nach dem Power-Up bzw. Reset des Mikrocontrollers ist das gesamte RELAYS-Byte „0“. Die Relais sind also im Start-/Ruhezustand deaktiviert.

Durch schreiben einer „1“ in eines der Bits (0..3) wird das entsprechende Relais aktiviert. Die LED auf dem PiXtend-Board leuchtet auf und das Relais schält hörbar.

Um die Funktion der Relais zu testen, kann für das RELAYS-Byte der Wert 255 (dezimal) bzw. 0xFF (hex) geschrieben werden. Alle vier Relais werden aktiviert. Dass dabei auch die Bits 4..7 gesetzt werden, hat keine Auswirkung.

2.2 GPIO_OUT

Bit	7	6	5	4	3	2	1	0
Name	**-**	**-**	**-**	**-**	**GPIO3**	**GPIO2**	**GPIO1**	**GPIO0**
Startwert	0	0	0	0	0	0	0	0

Bit 0..3

Die vier GPIO-Ausgänge sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von GPIO0. Bit 7 (MSB) bis Bit 4 sind nicht belegt.

Als Startwert nach dem Power-Up bzw. Reset des Mikrocontrollers ist das gesamte GPIO_OUT-Byte „0“. Die GPIOs sind im Start-/Ruhezustand deaktiviert.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temp./Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIO_CTRL. Weitere Informationen finden Sie in der App-Note für die Control- und Status-Bytes.

Durch schreiben einer „1“ in eines der Bits (0..3) wird der entsprechende GPIO aktiviert (falls als Ausgang konfiguriert). Falls der oder die GPIOs als Eingang oder für die genannten Sensoren konfiguriert ist, so hat das Verändern der Werte in GPIO_OUT keine Auswirkung.

Um die Funktion der GPIOs zu testen, müssen diese als Ausgänge konfiguriert werden. Anschließend kann für das GPIO_OUT-Byte der Wert 255 (dezimal) bzw. 0xFF (hex) geschrieben werden. Alle vier GPIO-Ausgänge werden aktiviert.
Dass dabei auch die Bits 4..7 gesetzt werden, hat keine Auswirkung.

2.3 PWMX (L/H)

Bei den PWM-Daten muss zwischen den beiden Betriebmodi "Servo-Mode" und "PWM-Mode" unterschieden werden. Die Umschaltung zwischen den Modi ist mit Hilfe der PWM_CTRL-Bytes möglich. Weitere Infos zu diesen Konfigurationsbytes finden Sie in einer separaten App-Note.

2.3.1 Servo-Mode

PWMXL - Servo-Value

Im Servo-Mode (Standard-Einstellung) steht in den PWMXL-Bytes der Wert für den Ausschlag eines Modellbauservos. Das „X“ ist durch die Nummer des PWM-Ausgangs zu ersetzen („0“ oder „1“).

Die Mittelstellung wird mit dem Wert 125 (dezimal) erreicht. Der Wert 0 entspricht dem Minimalausschlag, 250 dem Maximalausschlag.

Die Periodendauer des Signals beträgt 20 ms (50 Hz).

Minimalausschlag → 1 ms, Maximalausschlag → 2 ms

Die Start bzw. Initialwert des PWMXL-Bytes, nach einem Power-Up/Reset, ist „0“.

PWMXH - Servo-Overdrive-Value

Das PWMXH-Byte muss im Servo-Mode nicht zwingend benutzt werden. Das Byte wird nur dann ausgewertet, wenn der Overdrive für den entsprechenden Kanal aktiviert wurde.

Die Benutzung wird nur fortgeschrittenen Benutzern empfohlen!

Wird PWMXH auf den Wert 128 (dezimal) gesetzt, so hat der Overdrive keine Auswirkung (Mittelstellung). Jede Abweichung um +/-1 führt zu einer Veränderung um 4 Microsekunden, wie folgende Beispiele zeigen:

78 (-50)	→ **0,8 – 2 ms**
127 (-1)	→ 0,996 – 2 ms
128 (Mittelstellung)	→ **1 – 2 ms**
129 (+1)	→ 1 – 2,004 ms
178 (+50)	→ **1 – 2,2 ms**

Die meisten Modellbauservos kommen mit 0,8 – 2,2 ms „ON“-Zeit klar und fahren noch etwas weiter als im garantierten 1 – 2 ms Bereich. Tasten Sie sich vorsichtig an die Grenze heran.

Wird der Overdrive zu groß oder zu klein gewählt, so fährt der betroffene Servo auf Anschlag. Dies kann mechanische Defekte im Servo (Getriebe, Kunststoffteile usw.) zur Folge haben.

Die resultierende exzessive Stromaufnahme und Wärmeentwicklung kann die verwendeten Leitungen, die Stromquelle und den Servo beschädigen!

Folgende Reihenfolge sollte eingehalten werden um Probleme mit dem Overdrive zu vermeiden:

- PWM-Ausgang auf Servo-Mode konfigurieren (falls im PWM-Mode)
- Wert für den Servoausschlag in PWMXL schreiben (z.B. 125 für Mittelstellung)
- Wert für Overdrive auf Mittelstellung setzen (128)
- Overdrive für den Kanal aktivieren
- Overdrive langsam erhöhen und Servo über den gesamten Bereich bewegen (PWMXL von 0 bis 250 variieren)

Fährt der Servo beim Ausloten des maximalen Overdrive leicht auf Anschlag bzw. Getriebeende, so ist das in der Regel nicht schlimm. Der Servo beginnt zu brummen, da er nicht auf die geforderte Position fahren kann. Vermeiden Sie den Servo über längere Zeit in diesem Zustand zu belassen. Es kann zum oben genannten Defekt kommen und es wird ein erhöhter Strom aufgenommen.

2.3.2 PWM-Mode

PWMXL

Bit	7	6	5	4	3	2	1	0
Name								**LSB**
Startwert	0	0	0	0	0	0	0	0

PWMXH

Bit	7	6	5	4	3	2	1	0
Name	**MSB**							
Startwert	0	0	0	0	0	0	0	0

Im PWM-Mode enthalten die Bytes PWMXL und PWMXH ein zusammengehöriges 16 bit Datenwort. Dieses Datenwort wird für die Einstellung der Tastrate bzw. Duty-Cycle verwendet. Das „X“ wird durch die Nummer des jeweiligen PWM-Kanal ersetzt („0“ oder „1“). Jedem Kanal kann ein separater Duty-Cycle zugewiesen werden.

Welcher Duty-Cycle sich durch einen bestimmten Wert ergibt, hängt nicht ausschließlich von den PWMXL/H-Werten ab, sondern zusätzlich von den Control-Bytes „PWM_CTRL1“ und „PWM_CTRL2“. Weitere Infos hierzu sind der App-Note über Control- und Status-Bytes zu entnehmen.

Hier ein kleines Beispiel:

In PWM_CTRL1 und PWM_CTRL2 wird folgender 16 bit Wert geschrieben:
5000 (dezimal). In PWMXL/H wird der 16 bit Wert 2500 (dezimal) geschrieben. Der Duty-Cycle beträgt hiermit 50%.
Wird in PWMXL/H ein größerer Wert als in PWM_CTRL1/2 geschrieben, so verbleibt der PWM-Kanal durchgehend auf logisch „1“. Durchgehend logisch „0“ wird durch den Wert „0“ in PWMXL/H erreicht.

Wir empfehlen die Implementierung des PWM-Mode in Ihren eigenen Programmen zunächst ohne angeschlossene Geräte bzw. Aktoren zu testen und wenn möglich mit einem Oszilloskop zu überprüfen.

Setzen Sie die PWM-Werte mit bedacht!

Manche Aktoren können durch falsche Frequenzen oder Duty-Cycles beschädigt oder zerstört werden.
Überprüfen Sie Ihre Programme immer zunächst mit einem Oszilloskop, bevor echte Geräte an die PWM-Kanäle angeschlossen werden.

2.4 DIGITAL_IN

Bit	7	6	5	4	3	2	1	0
Name	**DI7**	**DI6**	**DI5**	**DI4**	**DI3**	**DI2**	**DI1**	**DI0**
Startwert	0	0	0	0	0	0	0	0

Die acht digitalen Eingänge (DIGITAL_IN) sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von DI0, das höchstwertige (MSB) den Zustand von DI7.

Die Eingänge haben im Ruhezustand immer den Wert „0“. Im aktiven Zustand (Spannung angelegt), wechselt der Wert auf „1“. Welcher Pegel zu einer „1“ führt, ist dem PiXtend-Datenblatt zu entnehmen.

2.5 ANALOG_INX (L/H)

ANALOG_INXL

Bit	7	6	5	4	3	2	1	0
Name								**LSB**
Startwert	0	0	0	0	0	0	0	0

ANALOG_INXH

Bit	7	6	5	4	3	2	1	0
Name	**-**	**-**	**-**	**-**	**-**	**-**	**MSB**	
Startwert	0	0	0	0	0	0	0	0

Der im PiXtend-Controller integrierte Analog/Digital-Wandler liefert 10 bit Werte. Die Werte sind in den Prozessdaten in einem 16 bit Datenwort, bestehend aus zwei Bytes, organisiert. Die höchstwertigen zwei Bits, des 10 bit Werts, stehen im Byte ANALOG_INXH.

Die digitalisierten Analogwerte können durch die Vorgabe von Wandlungsgeschwindigkeit und Mittelwertbildung beeinflusst werden. Die Konfiguration erfolgt über die Control-Bytes AI_CTRL0 und AI_CTRL1. Weitere Informationen finden Sie in der App-Note für die Control- und Status-Bytes.

In der Standard-Konfiguration, nach einem Reset oder Power-Cycle, läuft der A/D-Wandler mit einer Abtastrate von 125 kHz. Es werden jeweils 10 Werte aufgenommen, der Mittelwert gebildet und das Ergebnis in ANALOG_INX gespeichert.

Egal ob die Standard-Einstellung verwendet wird oder die AI_CTRL-Bytes manipuliert werden, die 16 bit ANALOG_INX-Bytes enthalten immer Werte zwischen „0“ und „1024“.

Um aus diesen Werten Spannungen und Ströme zu berechnen, werden folgende Berechnungsformeln verwendet:

Spannungen:

ANALOG_INX * 10 / 1024 = Spannung an Kanal X [V]

(Spannungen werden an AI0 und AI1 gemessen)

Bei den Spannungseingängen ist die Jumperstellung zu beachten. Die „10“ in der obigen Berechnungsformel wird in der Jumperstellung 10V verwendet.

Ströme:

ANALOG_INX * 0.02419411599 = Stromfluss bei Kanal X [mA]

(Ströme werden an AI2 und AI3 gemessen)

Der Umrechnungsfaktor ergibt sich aus Messwiderstand und Vorverstärkung (Operationsverstärkerschaltung).

Um die Genauigkeit der Kanäle zu nutzen, sollte bei der Berechnung mit Gleitkommazahlen (float/real) gearbeitet werden.

2.6 GPIO_IN

Bit	7	6	5	4	3	2	1	0
Name	**-**	**-**	**-**	**-**	**GPIO3**	**GPIO2**	**GPIO1**	**GPIO0**
Startwert	0	0	0	0	0/1	0/1	0/1	0/1

Die vier GPIO-Eingänge (GPIO_IN) sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von GPIO0. Bit 7 (MSB) bis Bit 4 sind nicht belegt.

Die GPIO-Eingänge haben keinen definierten Ruhezustand (floating input). Erst durch das externe anlegen einer Spannung ergibt sich ein stabiler Zustand:

0 V → Wert „0“

5 V → Wert „1“

Da die Eingänge schwimmend (floating) sind, können diese ohne externen Anschluss bei „0“ oder „1“ liegen bzw. zwischen den Werten hin und her wackeln.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temp./Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIO_CTRL. Weitere Informationen finden Sie in der App-Note für die Control- und Status-Bytes.

Nach einem Reset oder Power-Up von Pixtend sind die GPIOs als Eingänge konfiguriert.

2.7 TEMPX (L/H)

TEMPXL

Bit	7	6	5	4	3	2	1	0
Name								**LSB**
Startwert	0	0	0	0	0	0	0	0

TEMPXH

Bit	7	6	5	4	3	2	1	0
Name	**MSB**							
Startwert	0	0	0	0	0	0	0	0

Im DHT-Mode der GPIOs enthalten die Bytes TEMPXL und TEMPXH ein zusammengehöriges 16 bit Datenwort. Dieses Datenwort enthält die Temperatur-Information eines Sensors (falls Sensor angeschlossen und konfiguriert). Das „X“ steht für die Nummer des GPIOs, also 0 bis 3.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temp./Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIO_CTRL. Weitere Informationen finden Sie in der App-Note für die Control- und Status-Bytes.

Der enthaltende Wert kann sehr einfach in einen Temperaturwert (Gleitkommawert) umgewandelt werden. Es muss jedoch zwischen DHT11 und DHT22 unterschieden werden (TEMPX stellt den 16 bit Wert dar):

DHT11: TEMPX / 256 = Gleitkommawert [°C]

Beispiel: 5632 / 256 = 22,0 °C – *DHT11 Sensoren liefern keine Nachkommastelle!*

DHT22: TEMPX / 10 = Gleitkommawert [°C]

Beispiel: 224 / 10 = 22,4 °C

Je nach verwendeter Programmiersprache und Datentyp muss ein „typecast“ durchgeführt werden. Ansonsten ergibt sich nach dem Teilen durch 10 (DHT22) kein Gleitkommawert, sondern der ganzzahlige Wert (im Beispiel 22 statt 22,4).
Das Teilen durch 256 bei den DHT11-Sensoren kann auch durch einen „right shift“ um 8 Stellen erreicht werden.

2.8 HUMIDITY (L/H)

HUMIDITYXL

Bit	7	6	5	4	3	2	1	0
Name								**LSB**
Startwert	0	0	0	0	0	0	0	0

HUMIDITYXH

Bit	7	6	5	4	3	2	1	0
Name	**MSB**							
Startwert	0	0	0	0	0	0	0	0

Im DHT-Mode der GPIOs enthalten die Bytes HUMIDITYXL und HUMIDITYXH ein zusammengehöriges 16 bit Datenwort. Dieses Datenwort enthält die Luftfeuchtigkeits-Information eines Sensors (falls Sensor angeschlossen und konfiguriert). Das „X“ steht für die Nummer des GPIOs, also 0 bis 3.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temp./Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIO_CTRL. Weitere Informationen finden Sie in der App-Note für die Control- und Status-Bytes.

Der enthaltene Wert kann sehr einfach in einen Luftfeuchtigkeitswert (Gleitkommawert) umgewandelt werden. Es muss jedoch zwischen DHT11 und DHT22 unterschieden werden (HUMIDITYX stellt den 16 bit Wert dar):

DHT11: HUMIDITYX / 256 = Gleitkommawert [%] - relative Luftfeuchtigkeit

Beispiel: 10496 / 256 = 41,0 % – *DHT11 Sensoren liefern keine Nachkommastelle!*

DHT22: HUMIDITYX / 10 = Gleitkommawert [%] - relative Luftfeuchtigkeit

Beispiel: 417 / 10 = 41,7 %

Je nach verwendeter Programmiersprache und Datentyp muss ein „typecast“ durchgeführt werden. Ansonsten ergibt sich nach dem Teilen durch 10 kein Gleitkommawert, sondern der ganzzahlige Wert (im Beispiel 41 statt 41,7).
Das Teilen durch 256 bei den DHT11-Sensoren kann auch durch einen „right shift“ um 8 Stellen erreicht werden.

2.9 DIGITAL_OUT_READ

Bit	7	6	5	4	3	2	1	0
Name	**-**	**-**	**DO5**	**DO4**	**DO3**	**DO2**	**DO1**	**DO0**
Startwert	0	0	0	0	0	0	0	0

Diese Prozessdaten stehen nur im Manual-Mode und ab folgenden Mikrocontroller-Versionen zur Verfügung:

- PiXtend V1.2: Firmware-Version 12.6
- PiXtend V1.3: Firmware-Version 13.2

Das Byte DIGITAL_OUT_READ enthält den aktuellen Zustand der digitalen Ausgänge auf PiXtend. Damit wird die Überprüfung möglich, ob die digitalen Ausgänge tatsächlich gesetzt wurden oder nicht, was in manchen Anwendungen zweckmäßig sein kann.

Bit 0..5

Das Byte DIGITAL_OUT_READ hat den gleichen Aufbau wie das DIGITAL_OUT-Byte.

2.10 RELAYS_READ

Bit	7	6	5	4	3	2	1	0
Name	**-**	**-**	**-**	**-**	**RELAY3**	**RELAY2**	**RELAY1**	**RELAY0**
Startwert	0	0	0	0	0	0	0	0

Diese Prozessdaten stehen nur im Manual-Mode und ab folgenden Mikrocontroller-Versionen zur Verfügung:

- PiXtend V1.2: Firmware-Version 12.6
- PiXtend V1.3: Firmware-Version 13.2

Das Byte RELAY_READ enthält den aktuellen Zustand der Relais-Ausgänge auf PiXtend. Damit wird die Überprüfung möglich, ob die Relais tatsächlich gesetzt wurden oder nicht, was in manchen Anwendungen zweckmäßig sein kann.

Bit 0..3

Das Byte RELAY_READ hat den gleichen Aufbau wie das RELAY-Byte.