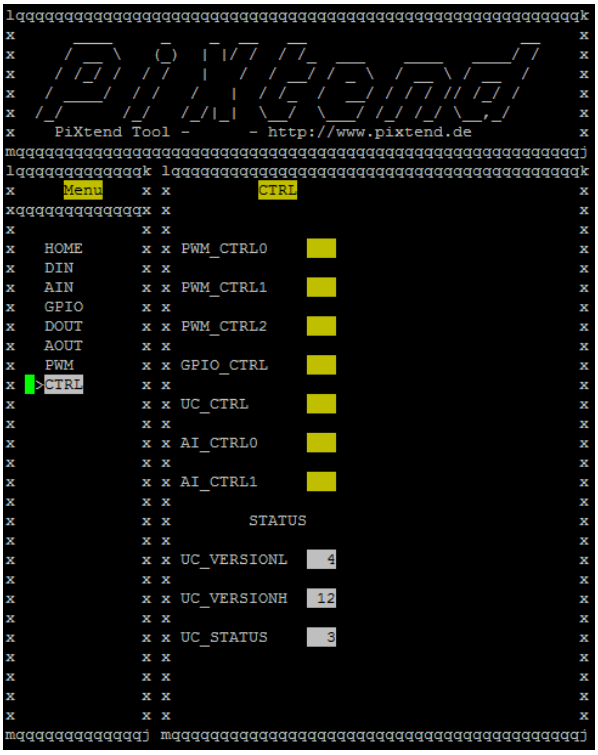
Application Note

PiXtend mit Hilfe der Control- & Status-Bytes konfigurieren und überwachen



Stand 21.02.2017, V1.06

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

Inhaltsverzeichnis

1. Einleitung & Allgemeines....3
1.1 Control-Bytes im Überblick....4
1.2 Status-Bytes im Überblick....4
2. Beschreibung der Control-Bytes....5
2.1 PWM_CTRL (0..2)....6
2.1.1 PWM_CTRL0....6
2.1.2 PWM_CTRL1..2....8
2.2 GPIO_CTRL....9
2.3 UC_CTRL....10
2.4 AI_CTRL (0..1)....12
2.4.1 AI_CTRL0....12
2.4.2 AI_CTRL1....13
2. Beschreibung der Status-Bytes....14
2.1 UC_VERSION (L/H)....14
2.2 UC_STATUS....14

1. Einleitung & Allgemeines

PiXtend bzw. der PiXtend-Mikrocontroller kann über Control-Bytes konfiguriert werden. Auf diese speziellen Bytes kann per Manual-Mode (*pixtendtool*), wie auch per Automatic-Mode (*pxauto* und *CODESYS*) zugegriffen werden.

Auf den folgenden Seiten wird zunächst ein Überblick über die Control- und Status-Bytes gegeben. Anschließend wird jedes Byte genau beschrieben. Es werden die unterschiedlichen Möglichkeiten aufgezeigt und Berechnungen durchgeführt.

Sollten trotzdem 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 Control-Bytes im Überblick

Es gibt folgende Control-Bytes:

- **PWM_CTRL (0..2)**

  Konfiguration der PWM-Kanäle: Umschaltung zwischen PWM- und Servo-Modus, Servo-Overdrive, Frequenz um Tastrate im PWM-Modus
- **GPIO_CTRL**

  Konfiguration der PiXtend-GPIOs: als Eingang, Ausgang oder für Temperatur und Luftfeuchtesensoren (DHT11/22, AM2302)
- **UC_CTRL**

  Grundlegende Konfiguration des PiXtend-Controllers (Watchdog, Betriebsmodus)
- **AI_CTRL (0..1)**

  Konfiguration der analogen Eingänge: Anzahl der Wandlungen (Mittelwertbildung), Abtastrate

1.2 Status-Bytes im Überblick

Es gibt folgende Status-Bytes:

- **UC_VERSION (0..1)**

  Enthält die Versionsnummer des PiXtend-Controllers.
- **UC_STATUS**

  Enthält den aktuellen Betriebsmodus des PiXtend-Controllers.

2. Beschreibung der Control-Bytes

Die Control-Bytes sind für die Konfiguration und Parametrierung des PiXtend-Mikrocontrollers zuständig. Die Standard-Einstellungen sind so gewählt worden, dass die Bedürfnisse der meisten Anwender befriedigt werden. Arbeiten Sie an den Control-Bytes, nur dann, wenn Sie sich über die Auswirkungen im Klaren sind. Die Control-Bytes werden nicht dauerhaft gespeichert. Nach einem Reset oder Power-Cycle sind die Werte wieder auf Standard-Werten.

Je nach Anwenderprogramm und angeschlossener Peripherie, kann es zu unerwartetem Verhalten kommen.

Machen Sie sich vor dem Verändern der Werte klar, wie sich diese auf Ihre Steuerungsanwendung und die angeschlossenen Aktoren, Sensoren oder andere Geräte auswirkt.

Die Control-Bytes können per *CODESYS*, dem *pixtendtool* und *pxauto* gesetzt werden.

2.1 PWM_CTRL (0..2)

2.1.1 PWM_CTRL0

Bit	7	6	5	4	3	2	1	0
Name	**CS2**	**CS1**	**CS0**	**-**	**-**	**OD1**	**OD0**	**Mode**
Startwert	0	0	0	0	0	0	0	0

Bit 0 – Mode

Das Mode-Bit entscheidet über den Betriebsmodus beider PWM-Ausgänge PWM0 und PWM1. Per Startwert „0“ sind die PWM-Ausgänge im Servo-Mode.

Wird Mode auf „1“ gesetzt, so wechseln beide Kanäle in den PWM-Modus

Servo-Mode:
Periodendauer 20 ms, 1 ms „ON“ Minimalausschlag, 2ms „ON“ Maximalausschlag Signal für Modellbauservos. Weitere Infos im PiXtend-Datenblatt.

PWM-Mode:
Frei einstellbare Periodendauer und Tastverhältnis (duty cycle). Einstellbar über CS0..2 und die beiden PWM_CTRL-Bytes 1 und 2.

Bit 1..2 – OD0..1

Mit den Overdrive-Bits wird im Servo-Mode entschieden ob der Overdrive aktiv „1“ oder inaktiv „0“ ist. Mit dem Overdrive lässt sich die „ON“ Zeit des Servo-Signals noch etwas vergrößern. Manche Servos können sich so noch etwas weiter drehen.
Der eigentliche Wert für den Overdrive wird ins Datenwort PWM0H und PWM1H geschrieben.

Zugehörigkeiten:

PWM0-Ausgang OD0 PWM0H

PWM1-Ausgang OD1 PWM1H

Bit 5..7 – CS0..2

Die Clock Select Bits (CS-Bits) ermöglichen die grobe Einstellung der PWM-Frequenz. Es handelt sich hier um die Prescaler-Bits des Timer 1 (16bit). Weitere Informationen finden Sie im Datenblatt des Mikrocontrollers (Atmel ATmega32A).

Die Einstellungen an den CS-Bits wirken sich nur im PWM-Mode aus.

Folgende Wahrheitstabelle gibt Aufschluss über die Auswirkungen der CS-Bits:

CS2	CS1	CS0	Beschreibung	Frequenz für Timer 1
0	0	0	Servo-Modus	250 kHz
0	0	1	Vorteiler (Prescaler): 1	16 MHz
0	1	0	Vorteiler (Prescaler): 8	2 MHz
0	1	1	Vorteiler (Prescaler): 64	250 kHz
1	0	0	Vorteiler (Prescaler): 256	62,5 kHz
1	0	1	Vorteiler (Prescaler): 1024	15,625 kHz

Wie die Frequenz / Periodendauer genau eingestellt werden kann, erfahren Sie bei den weiteren PWM_CTRL-Bytes.

2.1.2 PWM_CTRL1..2

Die PWM_CTRL-Bytes 1 und 2 enthalten ein zusammengehöriges 16 bit Datenwort. Dabei ist PWM_CTRL1 das Low-Byte und PWM_CTRL2 das High-Byte.

Mit den beiden Bytes kann die PWM-Frequenz / Periodendauer eingestellt werden. Die Periodendauer gilt für beide Kanäle. Die Tastrate kann für jeden Kanal separat mit den PWM_VALUEs eingestellt werden.

PWM_CTRL1

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

PWM_CTRL2

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

Beispiel für die PWM-Periodendauer:

Der Prescaler (CS0..2) wird auf den Wert 64 und damit auf 250kHz eingestellt. Außerdem wird der PWM-Mode aktiviert:

PWM_CTRL0: 01100001b

Das 16 bit Datenwort in PWM_CTRL1..2 wird auf den Wert 1000 eingestellt:

PWM_CTRL1: 11101000b

PWM_CTRL2: 00000011b

PWM-Periodendauer = Mikrocontroller-Frequenz / Prescaler / PWM_CTRL1..2

PWM-Periodendauer = 16 MHz / 64 / 1000 = ***250 Hz***

2.2 GPIO_CTRL

Bit	7	6	5	4	3	2	1	0
Name	**DHT3**	**DHT2**	**DHT1**	**DHT0**	**I/O3**	**I/O2**	**I/O1**	**I/O0**
Startwert	0	0	0	0	0	0	0	0

Bits 0..3 – I/O0..3

Mit den I/O-Bits können die vier PiXtend-GPIOs entweder als digitaler Eingang oder Ausgang konfiguriert werden. Mit dem Startwert „0“ sind alle GPIOs als Eingänge konfiguriert. Wird z.B. das Bit I/O3 auf „1“ gesetzt, so wird GPIO3 zum Ausgang. Der Ausgabewert wird über das Datenwort GPIO_OUT gesetzt.

Die GPIOs können außer Eingang/Ausgang auch für 1-Wire Sensoren (DHT11, DHT22, AM2302) genutzt werden. Dafür stehen die oberen vier Bits des GPIO_CTRL Datenworts bereit. Wird hier z.B. das Bit DHT3 auf „1“ gesetzt, so kann an GPIO3 ein solcher Sensor angeschlossen werden. Die Einstellung des I/O-Bit spielt dann keine Rolle mehr. Die DHTX-Bits setzen sich gegen die I/OX-Bits durch.

Die Ergebnisse/Messwerte der DHT-Sensoren stehen in TEMPXL/TEMPXH und HUMIDITYXL/HUMIDITYXH.

2.3 UC_CTRL

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

Bit 0 – WD

Mit dem WD-Bit kann der Watchdog des PiXtend-Mikrocontroller aktiviert werden. Dieser ist standardmäßig ausgeschaltet (Wert „0“). Zum aktivieren des Watchdogs wird eine „1“ geschrieben.

Der Watchdog ist für den Automatic-Mode gedacht. Ist WD aktiviert, wird die Kommunikation zwischen Raspberry Pi und PiXtend überwacht. Gibt es zwischen zwei Automatic-Mode-Zyklen eine Verzögerung größer als 2 Sekunden, so schlägt der Watchdog zu und führt einen Reset des Mikrocontrollers durch. Dieses Verhalten kann dann gewünscht werden, wenn PiXtend in einen sicheren bzw. definierten Zustand gesetzt werden soll, wenn sich die Software auf Seiten des Raspberry Pi aufhängt, falsche Daten sendet oder die Übertragung gestört wird.

Bit 4 – RUN

Auch das RUN-Bit ist nur für den Automatic-Mode gedacht. Es handelt sich hier um eine Freischaltung bzw. ein Handshake zwischen Raspberry Pi und Mikrocontroller. Im Startzustand steht das RUN-Bit auf „0“. Der Mikrocontroller kommuniziert zwar mit dem Raspberry, doch werden die Daten nicht weiter ausgewertet, also keine Eingangswerte zurückgegeben oder Ausgänge gesetzt. Es muss zunächst folgender Handshake abgearbeitet werden, um den Mikrocontroller vom INIT-State in den RUN-State zu versetzen:

- Mikrocontroller startet nach dem Power-Up oder Reset im INIT-State (RUN-Bit ist auf „0“ gesetzt).
- UC_CTRL, welches bei jedem Automatic-Mode-Cycle übertragen wird, wird im Anwendungsprogramm (auf dem Raspberry Pi) eingestellt. Das RUN-Bit wird auf „1“ gesetzt.
- Der Mikrocontroller erkennt die Anforderung, vom INIT-State in den RUN-State zu wechseln. Wenn kein Fehler (CRC-Prüfsumme) vorliegt, so wechselt der Controller sofort in den RUN-State und die eigentliche Kommunikation wird aufgenommen.

- Zusätzlich signalisiert der Mikrocontroller über das Status-Wort UC_STATUS, dass er sich nun im RUN-State befindet.
- Auf der Seite der Raspberry-Software, kann die Anforderung (RUN-Bit) wieder zurückgenommen, also auf „0“ gesetzt werden.

Der ganze Ablauf hat den Sinn, dass die Verarbeitung der Kommunikationsdaten zwischen Mikrocontroller und Raspberry Pi nach einem Fehler (CRC-Error, Watchdog abgelaufen) nicht direkt und unbemerkt wieder anläuft, als wäre kein Fehler passiert.

Wird der Ablauf wie oben Beschrieben durchgeführt, so bleibt der Mikrocontroller, nach einem Fehler, in einem definierten Zustand. Erst nach der expliziten Anforderung des RUN-State, werden wieder alle Daten ausgetauscht.

2.4 AI_CTRL (0..1)

2.4.1 AI_CTRL0

Bit	7	6	5	4	3	2	1	0
Name	**NoS3_1** (AI3)	**NoS3_0** (AI3)	**NoS2_1** (AI2)	**NoS2_0** (AI2)	**NoS1_1** (AI1)	**NoS1_0** (AI1)	**NoS0_1** (AI0)	**NoS0_0** (AI0)
Startwert	0	0	0	0	0	0	0	0

Mit den Bits in AI_CTRL0 kann die Anzahl der Messungen für jeden analogen Eingang konfiguriert werden. Bei der Standard-Einstellung „0“ werden für jeden analogen Eingang jeweils 10 Messungen durchgeführt, anschließend eine Mittelwertbildung durchgeführt und erst dann an den Raspberry Pi weitergegeben. Die Verarbeitung der Mittelwerte läuft komplett unbemerkt im Mikrocontroller ab.

Die Anzahl der Messungen für eine Mittelwertbildung wird für jeden Kanal nach dem selben Prinzip eingestellt, wie folgender Wahrheitstabelle zu entnehmen ist:

NoSX_1	NoSX_0	Beschreibung
0	**0**	**Standard: 10 Messungen**
0	**1**	**1 Messung (keine Mittelwertbildung)**
1	**0**	**5 Messungen**
1	**1**	**50 Messungen**

NoS bedeutet „Number of Samples“. Die darauf folgende Nummer gibt den analogen Eingang an, auf den sich die Einstellung auswirkt. Die letzte Ziffer unterscheidet noch zwischen den zwei Bits, welches es für jeden Kanal gibt.

Je mehr Messungen gemacht werden, umso weniger Rauschen enthalten die Signale. Wenn es also auf die Genauigkeit ankommt, empfehlen wir 10 oder 50 Messungen für jede Mittelwertbildung einzustellen.

2.4.2 AI_CTRL1

Bit	7	6	5	4	3	2	1	0
Name	**CS2**	**CS1**	**CS0**	**-**	**-**	**-**	**-**	**-**
Startwert	0	0	0	0	0	0	0	0

Die CS-Bits (Clock Select) beeinflussen die Abtastrate des A/D-Wandlers, welcher für alle analogen Eingänge zuständig ist. Wird eine Einstellung vorgenommen, wirkt sich diese also auf alle vier Eingänge gleichermaßen aus.
Auch hier entsprechen die Bits dem Clock Select / Prescaler des Mikrocontroller-ADCs. Weiter Informationen erhalten Sie im Datenblatt des Atmel ATmega32A.

Der Startwert ist für alle drei Bits „0“.

Folgende Wahrheitstabelle gibt Aufschluss über die Auswirkungen der CS-Bits:

CS2	CS1	CS0	Beschreibung	Frequenz für den A/D-Wandler
0	**0**	**0**	**Standard - Vorteiler (Prescaler): 128**	**125 kHz**
0	**0**	**1**	**Vorteiler (Prescaler): 2**	**8 MHz**
0	**1**	**0**	**Vorteiler (Prescaler): 4**	**4 MHz**
0	**1**	**1**	**Vorteiler (Prescaler): 8**	**2 MHz**
1	**0**	**0**	**Vorteiler (Prescaler): 16**	**1 MHz**
1	**0**	**1**	**Vorteiler (Prescaler): 32**	**0,5 MHz**
1	**1**	**0**	**Vorteiler (Prescaler): 64**	**250 kHz**

Je größer die Frequenz, umso ungenauer bzw. verrauschter ist das Ergebnis. Falls keine besonderen Anforderungen an die Geschwindigkeit gestellt werden, so empfehlen wir die Standard-Einstellung beizubehalten.

3. Beschreibung der Status-Bytes

Die Status-Bytes informieren den Anwender bzw. das Anwenderprogramm, welches auf dem Raspberry Pi läuft, über den Status des Mikrocontrollers.

Die Status-Bytes können per CODESYS, dem pixtendtool und pxauto abgefragt werden.

3.1 UC_VERSION (L/H)

Die UC_VERSION-Bytes geben die Mikrocontroller/PiXtend-Version wieder.

UC_VERIONL: Versionsstand der Mikrocontroller Firmware
Ausgabewert (Beispiel): 3 (dezimal), entspricht der Firmware-Version 3

UC_VERSIONH: Versionsstand der PiXtend-Leiterplatte
Ausgabewert (Beispiel): 13 (dezimal), entspricht der Leiterplattenversion 1.3

3.2 UC_STATUS

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

Bit 0 – State

Das State-Bit gibt den Zustand des Mikrocontrollers im Automatic-Mode wieder. Beim Wert „0“ befindet sich der Mikrocontroller im INIT-State, beim Wert „1“ im RUN-State. Erst im RUN-State werden gültige Daten ausgetauscht. Es kann also das RUN-Bit überwacht werden, um die Gültigkeit der Daten zu überprüfen. Weitere Infos zum RUN/INIT-State finden sich bei der Beschreibung des UC_CTRL-Byte auf Seite 10.
Für den Manual-Mode, welcher z.B. beim pixtendtool Verwendung findet, spielt das State-Bit keine Rolle und muss nicht beachtet werden.

Bit 1 – Debug

Das Debug-Bit informiert darüber, ob der PiXtend-GPIO0 als DEBUG-Pin verwendet wir oder nicht. Eine „0“ bedeutet, dass der GPIO0 als normaler Eingang, Ausgang oder für DHT-Sensoren eingesetzt wird (je nach Einstellung des GPIO_CTRL-Byte). Eine „1“ signalisiert, dass das GPIO0 als Debug-Pin verwendet wird. Einstellungen im GPIO_CTRL-Byte für GPIO0 haben so keine Auswirkung auf die Funktion.

Das aktivieren oder deaktivieren der Debug-Funktion ist nur im Mikrocontroller-Quelltext möglich und sollte nur von fortgeschrittenen Benutzern geändert werden. Standardmäßig ist die Debug-Funktion deaktiviert.